VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Grafické uživatelské rozhraní pro správu cestovních profilů pomocí MS Office Profile Wizard

Graphical User Interface for Managing Travel Profiles Using MS Office Profile Wizard

2010 Bc. Petr Hrabec

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava*.

V Ostravě 1. května 2010 .....

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 1. května 2010 .....

Abstrakt

Cílem práce bylo seznámení se s problematikou cestovních profilů. Vytvoření nového programu, jehož funkčnost je totožná s MS Office Profile Wizard a srovnání rychlostí práce obou programů. Vytvoření uživatelského rozhraní ke konfiguračnímu souboru pro MS Office Profile Wizard. V práci jsou uvedeny ukázky, algoritmy a jiná konkrétní řešení nezbytná pro chod vytvořených aplikací.

Klíčová slova: C#, .NET, MS Office Profile Wizard, Windows Server, WinForms

Abstract

Goal of work was identification with roaming profiles. Develop new program, based on MS Office Profile Wizard and compare their performance. Develop user interface for INI file for MS Office Profile Wizard. There are samples, algorithms and other concrete solutions which are nessesary for running our applications.

Keywords: C#, .NET, MS Office Profile Wizard, Windows Server, WinForms

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Diplomová práce se zabývá potřebami firmy ORBIT s.r.o., která používá k uchování nastavení profilů jednotlivých uživatelů v doméně cestovní profily.

Cílem diplomové práce je seznámit čtenáře s problematikou cestovních profilů (roaming profiles), kde důležitou roli hraje správné nastavení politik, ukázky jednotlivých kroků vedoucí k správnému vytváření profilů a pár rad či zajímavostí k této problematice.

Dále si představíme implementaci programu, který kopíruje práci programu MS Office Profile Wizard a srovnáme si rychlosti s jakými oba programy pracují.

Nakonec se podíváme jak obsloužit program MS Office Profile Wizard, respektive sestavení konfiguračního souboru, na jehož základě se řídí práce programu MS Office Profile Wizard.

2 Analýza potřeb firmy

2.1 Hlavní myšlenka

Firma ORBIT s.r.o. [1] používá k uchování nastavení profilů jednotlivých uživatelů v doméně cestovní profily. Díky tomu se mohou uživatelé přihlašovat do domény odkudkoliv při zachování jejich nastavení.

Každodenní práce má za následek, že objem uživatelského profilu každým dnem narůstá. Kvůli zvětšujícímu se objemu dat dochází k prodlužování doby přihlášení uživatele k počítači. Je třeba najít způsob, kterým se urychlí přihlašování uživatelů do domény.

2.2 Řešení

2.2.1 Seznámení s MS Office Profile Wizard

Nástrojem, který by pomohl urychlit přihlašování uživatelů do domény je MS Office Profile Wizard, viz také [8-10]. Tato aplikace vytváří ze seznamu souborů a registrů datový soubor, ze kterého lze pak tyto soubory a registry bez problému obnovit. Jeho využití je prosté. S rychlostí jakou MS Office Profile Wizard vytváří datový soubor, či obnovuje z něj data, urychlí a zastoupí nahrávání souborů profilu ze serveru.

Administrátor, který bude mít na starosti správu těchto profilů, musí nastavit, které soubory a registry se budou ukládat a obnovovat pomocí MS Office Profile Wizard. K tomu slouží konfigurační soubor aplikace. Pro každou skupinu uživatelů, či pro jednotlivce lze zadat individuální nastavení. Toto nastavení použije MS Ofice Profile Wizard a vyexportuje soubory a registry z nastavení do vybraného uložiště. Aby bylo toto řešení co nejefektivnější, po vytvoření se datový soubor ještě zkomprimuje a uloží se do úložiště v zakomprimované podobě.

2.2.2 Použití aplikace

Nejideálnější dobou pro použití aplikace je spustit export dat při odhlašování uživatele. Není to však nutnou podmínkou, ale může nastat problém, že uživatel bude pracovat na souboru, který je v seznamu pro export a soubor by nebyl úspěšně exportován. V tomto případě by musel práci na tomto souboru ukončit. Pro obnovu dat je nejideálnější dobou použití aplikace při přihlašování uživatele do domény.

MS Office Profile Wizard disponuje několika režimy, ve kterých pracuje. Důležitou funkcí a předností aplikace je, že dokáže pracovat, aniž by o tom uživatel věděl, a navíc nezabírá tolik procesorového času a paměti.

Konfigurace argumentů pro spuštění aplikace pro vytvoření datového souboru:

- /q - tichý mód, spuštění aplikace bez indikátoru postupu a chybových zpráv
- /i INI soubor - cesta a název konfiguračního souboru
- /s OPS soubor - cesta a název datového souboru pro uložení

Samotný příkaz by vypadal následovně: PROFLWIZ.EXE /q /i custom.ini /s custom.ops

Konfigurace argumentů pro spuštění aplikace při obnově dat z datového souboru:

- /q - tichý mód, spuštění aplikace bez indikátoru postupu a chybových zpráv
- /d - Reset to defaults funkce - vymaže soubory a regstry před obnovou
- /i INI soubor - cesta a název konfiguračního souboru
- /r OPS soubor - cesta a název datového souboru pro obnovu

Samotný příkaz by vypadal následovně: PROFLWIZ.EXE /q /d /i custom.ini /r custom.ops

2.3 Zhodnocení

Po uvedeném řešení lze použitím MS Office Profile Wizard značně urychlit načítání uživatelského profilu při přihlašování. Konfigurační soubor aplikace lze sestavovat na základě reálných hodnot na uživatelských účtech. Aby administrátor měl práci sestavování konfiguračního souboru ulehčenou, bude mít k dispozici aplikaci uživatelského rozhraní pro konfigurační soubor MS Office Profile Wizard. Díky této aplikace může vytvářet nové či upravovat stávající konfigurační soubory. Popis práce a použití aplikace je uveden dále v textu.

3 Problematika cestovních profilů

Cestovní profily jsou jednou z mnoha funkcí, kterými disponuje Windows Server 2003, na kterém je tato problematika odzkoušena. Nedílnou součástí je také problematika domén, doménových řadičů a Active Directory. Informace k těmto prvkům jsou k nalezení zde [2-6].

3.1 Cestovní profily

Cestovní profily slouží k uchování nastavení profilů "Documents and Settings" jednotlivých uživatelů v doméně. Tyto profily bývají zpravidla ukládány na server (doménový řadič). Cestovní profily slouží převážně k tomu, aby profil uživatele byl stále s ním, i když se do domény přihlásí přes jiný počítač, protože při přihlášení se mu jeho osobní profil stáhne ze serveru.

Cestovní profily fungují jako synchronizace. Když se uživatel přihlásí do domény, porovnají se obě verze profilů (ta na serveru a ta na počítači v síti), a když lokální kopie profilu bude novější, přepíše se cestovní profil na serveru. Je to z toho důvodu, že ne vždy je uživatel připojen k síti a provádí změny v profilu. Při odhlášení se pak změny provedené v profilu ukládají do cestovního profilu na serveru.

Když uživatel není v síti a přihlašuje se do domény, systém na toto upozorní a načte se mu lokální kopie profilu.

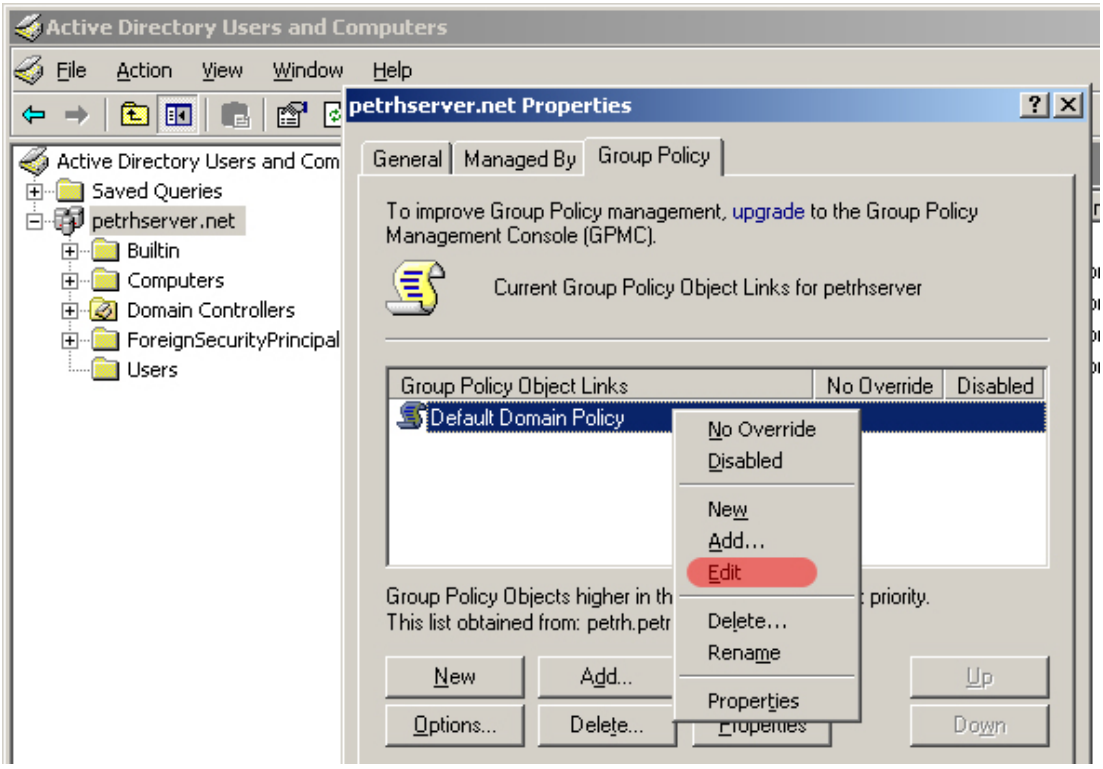
Při prvním přihlášení uživatele do domény se načte uživateli tzv. defaultní profil, který je společný pro všechny prvně se přihlášející uživatele.

3.1.1 Vytvoření složky pro profily

Prvním krokem k vytváření cestovních profilů je vytvoření sdílené složky na serveru (název složky nerozhoduje, většinou se pojmenovává "Profiles"). Důležitým nastavením u této sdílené složky, ve vlastnostech na kartě "Sharing" v sekci "Permissions", je, že skupina "Everyone" bude mít "full control". Jedná se o oprávnění navenek (sdílení do sítě). Další důležitou kartou ve vlastnostech sdílené složky "Profiles" je "Security". Jedná se zpravidla o NTFS oprávnění. Je dobré se ujistit, že skupina "Domain users" mají také "full control". Docházíme tak k vazbě mezi kartami "Sharing" a "Security".

3.1.2 Vazba mezi "Sharing" a "Security"

Mezi kartami "Sharing" a "Security" je tzv. logický součin. Ve většině případech se nastavují všechna práva v "Sharing" a reguluje se to na kartě "Security". Př.: Na kartě "Sharing" dáme práva pro administrátora, tím mu zajistíme v rámci sítě zasahovat do sdílené složky, pokud ale nenastavíme pro něj práva na kartě "Security", nebude mít v konečné fázy do skožky přístup, protože logický součin 1 a 0 je 0. V případě nastavení práv na kartě "Security" zajistíme adminisrátorovi přístup do složky, protože logický součin 1 a 1 je 1.



Obrázek 1: Nastavení politky pro složku cestovního profilu

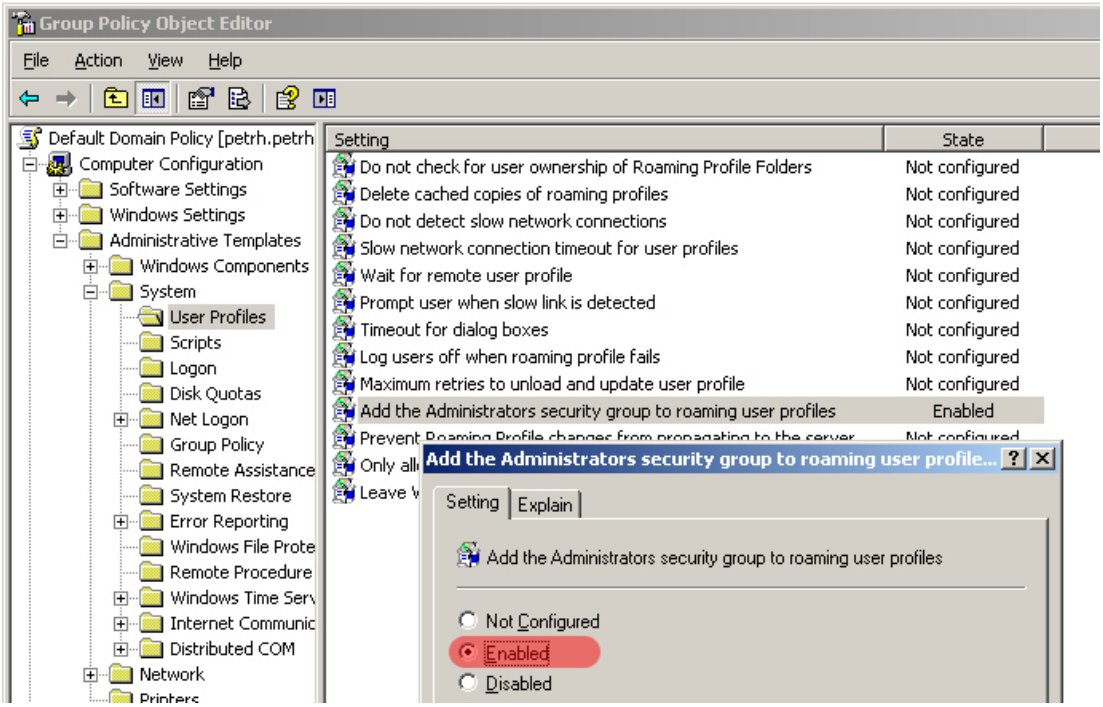
3.1.3 Složka cestovního profilu

Po správném nastavení sdílené složky "Profiles" je třeba implicitně zajistit nově vytvořeným cestovním proflům jiné než defaultní nestavení. Jelikož při vytvoření cestovního profilu má právo do domovské složky toho profilu pouze ten uživatel, který složku vytvořil, ne administrátor. Buď se přidává ručně (což při velkém množství profilů je neefektivní), nebo nastavením politiky, tzn. že se při každém nově vytvořeném cestovním profilu přidá také administrátor.

3.1.4 Nastavení politky pro složku cestovního profilu

Přístup do nastavení politiky se realizuje pomocí konzole "Group Policy Object Editor". Pokud, jako v tomto případě, není v administrátorském nástroji tato konzole obsažena, alternativa se pak nachází ve vlastnostech domény v konzoli "Active Directory Users and Computers" na kartě "Group Policy". Zde se nachází defaultní politika, kde pod položkou edit se nachází konzole "Group Policy Object Editor" viz obrázek 1.

K samotnému nastavení politiky vede cesta přes "Computer Configuration" -> "Administrative Templates" -> "System" -> "User Profiles". Pod složkou "User Profiles" se nachází nastavení "Add the Administrators security group to roaming user profiles", které je potřeba nastavit na "Enabled". Tím zajistíme, aby také administrátoři měli



Obrázek 2: Přístup administrátorů k cestovnímu profilu

přístup do cestovních profilů uživatelů, viz obrázek 2. Pozn.: Vypnutí tohoto nastavení se realizuje nastavením ”Disabled” ne ”Not Configured”!

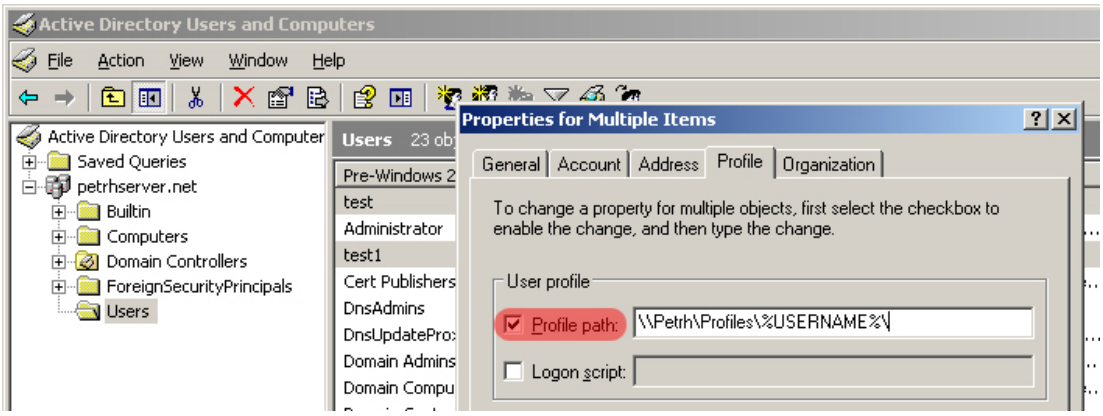
3.1.5 Nastavení cestovních profilů

Aby profil uživatele byl registrován jako cestovní, je třeba tuto možnost zaznačit ve vlastnostech uživatele na kartě ”Profile”, viz obrázek 3.

V praxi se převážně využívá možnost tohoto nastavení hromadně. Cesta k profilu se nastavuje pomocí UNC cesty, tudíž síťové cesty, jak je profil vidět z venku ze sítě. Pokud se dodržuje standartů, cesta k profilu se zapisuje: ”\Jméno serveru\Složka cestovních profilů\%USERNAME%\”. Pod %USERNAME% se nachází přihlašovací jméno uživatelů. Je logické mít přehled nad názvy složek, protože nastávají situace, že je třeba individuálně upravit profil uživatele, a když nebudeme mít nad tímhle přehled, onoho profilu se dohledat nemusíme.

3.2 Soubor NTUSER.DAT

Jednou z používaných možností v praxi, je úprava přípony souboru NTUSER.DAT v profilu uživatele, ve kterém jsou uloženy registry profilu. Používá se to proto, že pokud nechceme aby byly uloženy změny v profilu, tak v defaultnímu profilu přepíšeme příponu souboru NTUSER.DAT na NTUSER.MAN.



Obrázek 3: Nastavení profilu jako cestovní

4 New profile wizard

4.1 MS Office Profile Wizard

MS Office Profile Wizard (dále jen PW) slouží k ukládání a k obnově Microsoft Office souborů nebo registrů, které si sám uživatel nadefinuje. Tento fakt lze rozšířit i tím, že nejsme omezení typem souboru, tudíž můžeme využít PW i k ukládání jiných než Microsoft Office souborů.

K nastavování souborů nebo registrů, které se uloží do datového souboru (které pak budou při obnově z tohoto souboru obnoveny), slouží INI soubor. Tento konfigurační soubor nabízí dva typy nastavení: pro soubory a pro registry. Každý z těchto typů pak obsahuje několik možností (dále jen vlastností) jak s těmito typy naložit.

Úkolem aplikace New Profile Wizard (dále jen NPW) je, aby kopíroval práci PW, a to co nejefektivněji.

4.2 New profile wizard

NPW je napsán na platformě .NET Framework 2.0 jazykem C# 2.0 a je vytvořen jako Windows Forms Aplikace.

4.2.1 Čtení INI souboru

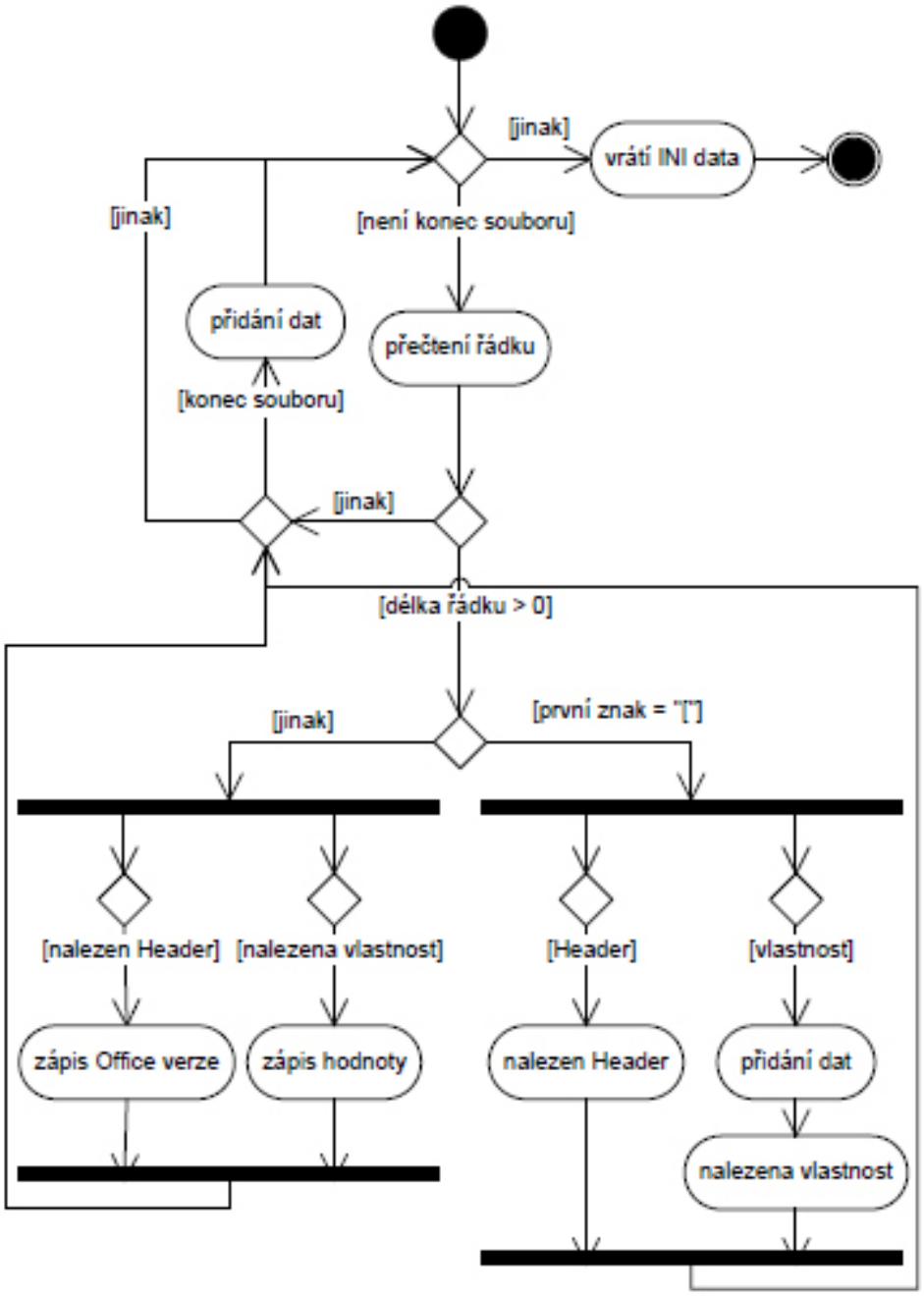
NPW pracuje tak, že parsuje konfigurační neboli INI soubor řádek po řádku, dokud nenarazí na konec souboru. Styl a syntaxe psaní řádků je definován v samotném INI souboru tak, jak ukazuje výpis 1.

```
[ IncludeIndividualFiles ]
# List individual files to be included into the OPS file.
# Syntax is one path\filename per line.
# Entries must begin with one of the Folder tokens listed under
#   [IncludeFolderTrees].
# Wildcards are not supported.
#
# Example for including Normal.dot:
#   <AppData>\Microsoft\<SubFolder_Templates>\Normal.dot # word
```

Výpis 1: Ukázka definice syntaxe řádku určité vlastnosti

NPW ignoruje řádky typu komentářů, prázdných řádků apod., tzn. že definice syntaxe na výpisu 1 je vlastně syntaxí komentáře, tudíž bude v parsování INI souboru ignorována. Řádky, se kterými NPW pracuje, jsou dva druhy: vlastnosti a jejich hodnoty. Vlastnosti se zapisují do hranatých závorek. Vlastností je daný počet a každá vlastnost má unikátní název. Když narazí NPW na platnou vlastnost, začne číst hodnoty této vlastnosti, dokud nenarazí na další platnou vlastnost. Aktivitní diagram na obrázku 4 zjednodušeně popisuje průběh čtení INI souboru.

Tyto hodnoty nemusí mít správnou syntaxi, mohou obsahovat nesprávně zadané souborové tokeny (viz. sekce Sestavování absolutních cest), respektive řádek se zapsanou



Obrázek 4: Aktivitní diagram čtení INI souboru

hodnotou nemusí mít správnou vypovídající hodnotu pro NPW, tudíž tyto řádky NPW eliminuje ještě ve fázi čtení INI souboru a ignoruje je.

Relativně správné získané hodnoty se ukládají do objektu typu Dictionary<string, List<string>>, kde klíčem slovníku je název vlastnosti a hodnotou pod klíčem je seznam hodnot této vlastnosti. Díky takovému uspořádání dostaneme totožnou kopii obsahu INI souboru do jednoho objektu, který je přehlednou startovací pozicí pro další zpracování.

Speciálním případem, který potřebujeme při parsování konfiguračního souboru odchytit, je poznamenat si verzi MS Office, která je na uživatelském účtě nainstalována. Číslo verze se nachází za řádkem "[Header]" a slouží pro rozpoznání speciálního druhu cesty k MS Office souborům.

4.2.2 Sestavování absolutních cest

Zápis hodnot v INI soboru, respektive cest k souborům, se kterými má NPW pracovat, se skládá z tzv. souborových tokenů a relativní cesty k souboru. Token je označení složky v uživatelském profilu v "Documents and Settings". Seznam tokenů odpovídá základním složkám v profilu. Tokeny, které jsou tedy povoleny jsou:

- <AppData>
- <Desktop>
- <Favorites>
- <Personal>
- <ProgramsMenu>
- <RecentFiles>
- <SendTo>
- <StartMenu>
- <StartupMenu>
- <UserProfile>

Důvod, proč se používají tokeny, je velice jasný. Ne každý systém má stejnou cestu k profilu uživatele, ať už se jedná o lokalizaci systému, nebo jenom různé názvy disků, na kterých je systém nainstalovaný, tudíž je zapotřebí vyhledat tyto absolutní cesty na základě zadaných tokenů.

Absolutní cesty k těmto adresářům lze najít v hodnotách registru HKEY_CURRENT_USER pod klíčem Software\\Microsoft\\Windows\\CurrentVersion \\Explorer\\Shell Folders. Těmito cestami pak NPW nahradí souborový token a vytvoří tak absolutní cestu k zadanému souboru.

Jedním speciálním případem jsou tokeny <Subfolder_Název hodnoty>, které se používají, když je zapotřebí do relativní cesty k souborovému tokenu přidat složku, která

má opět jiný název například kvůli jazykové verze systému. Tyto hodnoty se nacházejí pod klíčem HKEY_CURRENT_USER\\Software\\Microsoft\\Office\\"Číslo verze MS Office"\\Common\\General. Jak je vidět z této cesty, je jasné k čemu je zapotřebí při parsování konfiguračního souboru poznamenat číslo verze MS Office. Toto číslo nám bude tedy sloužit jako identifikátor verze a při hledání hodnot k tokenu Subfolder bude NPW hledat tyto hodnoty pod správnou cestou klíče registru.

Co se týče hodnot pod vlastnostmi týkajících se registrů, tak jejich zápis v této fázi nepotřebuje dalšího zpracování.

Po těchto upravávách se tedy původní objekt změnil pouze v seznamu hodnot tak, že hodnoty vlastností souborů dostaly konkrétní podobu pro daný systém.

Nyní se práce NPW rozděluje do dvou větví: zapisování dat do datového souboru a obnova dat z něj. Obě dvě operace vyžadují předpřipravená data z INI souboru ve formě výše zmiňovaného objektu.

4.2.3 Zápis dat do datového souboru

Vytváření seznamu souborů

NPW zapisuje do datového souboru všechny soubory, které byli uvedeny v INI souboru, a jeden soubor registrů, který byl vytvořen ze seznamu registrů.

Seznam souborů, které se zahrnou do datového souboru, je vytvořen z hodnot uvedených pod vlastnostmi:

- IncludeFolderTrees
- IncludeIndividualFolders
- IncludeIndividualFiles

Tento seznam souborů se ještě redukuje podle hodnot uvedených pod vlastnosmi ExcludeFiles. Ve skutečnosti to znamená, že vytvořený seznam se musí redukovat na základě hodnot v ExcludeFiles, protože soubory zde uvedené nebudou do datového souboru zahrnuty, tudíž je musíme z původního seznamu odstranit.

Důvod, proč vlastnost ExcludeFiles existuje, je takový, že soubory z hodnot ve vlastnostech IncludeFolderTrees a IncludeIndividualFolders jsou dynamicky načteny, tudíž nemáme přehled nad každým souborem zvlášť. A proto výjimky, které zahrnout do datového souboru nechceme, NPW odstraní.

Pojmem dynamicky načtené soubory se rozumí načíst seznam souborů z uvedeného adresáře. V připadě IncludeIndividualFolders to je jednoduší, protože stačí použít třídu Directory.GetFiles(název adresáře). V případě IncludeFolderTrees už to tak jednoduché není, protože předem neznáme stromovou strukturu adresářů, proto seznam adresářů musíme získat rekurzivně, viz. výpis 2. Jakmile NPW zjistí všechny podadresáře uvedeného adresáře v IncludeFolderTrees, může na něj aplikovat výše uvedenou metodu třídy Directory a získat tak všechny soubory v daném adresáři. Pokud adresář nebo soubor, který byl uveden jako hodnota vlastností neexistuje, tuto anomálii NPW zachytí

pomocí metody Exists tříd File a Directory a tuto hodnotu ignoruje.

```
// parametry: dir – název adresáře, ze kterého se budou získávat podadresáře, theList – seznam
      aktuálně získaných adresářů
void GetDirectorySubDirectories(string dir, List<string> theList)
{
    if (Directory.GetDirectories(dir).Length > 0)
    {
        theList.Add(dir);
        for (int i = 0; i < Directory.GetDirectories(dir).Length; i++)
        {
            GetDirectorySubDirectories(Directory.GetDirectories(dir)[i], theList);
        }
    }
    else
    {
        theList.Add(dir);
    }
}
```

Výpis 2: Rekurzivní procházení adresářové struktury

Wildcards

Speciálním případem zápisu hodnot je použití tzv. wildcards, což jsou znaky, které nám nahrazují určitou nebo celou část názvu. Znaky, které nahrazují části názvu jsou "*" - představuje jakýkoliv řetězec znaků a "?" - představuje jakýkoliv znak.

Konkrétně se wildcards používají v ExcludeFiles v názvech souboru. Př. pro zápis souboru Normal.dot: Normal.* nebo Norm??.dot. K použití wildcards má NPW třídu Wildcard.cs, která dědí z třídy Regex, kde vstupní řetězec např. "Norm??.dat" je upraven na "Norm..\\.dot^$", což je tvar regulárního výrazu. Metoda, která provádí tuto úpravu, se nachází na výpisu 3, viz také [7,15,16]. Tímto tvarem regulárního výrazu můžeme porovnávat pomocí metody IsMatch(název souboru) názvy soborů. Pokud tedy najdeme v seznamu souboru název souboru, který odpovídá dané wildcard, tak bude z onoho souboru vyjmut.

```
public static string WildcardToRegex(string pattern)
{
    return "^" + Regex.Escape(pattern).Replace("\\*", ".*").Replace("\\?", ".") + "$";
}
```

Výpis 3: Metoda převádějící řetězec s wildcard na regulární výraz

Soubor registrů

Myšlenka vytvoření souboru registrů vychází z vytváření seznamu souborů pro datový soubor. Opět existují vlastnosti, které zahrnují klíče registrů a jejich hodnoty do souboru registrů jako:

- IncludeRegistryTrees
- IncludeIndividualRegistryKeys
- IncludeIndividualRegistryValues

Dále také existují vlastnosti, které tento seznam opět redukují:

- ExcludeRegistryTrees
- ExcludeIndividualRegistryKeys
- ExcludeIndividualRegistryValues

Stejně se načtou všechny klíče a jejich vlasnosti dynamicky, a v případě IncludeRegistryTrees opět rekurzivně, viz. výpis 4, a stějně se pak konkrétní hodnoty nebo klíče z tohoto seznamu odstraní.

```
// parametry: dir – název adresáře, ze kterého se budou získávat podadresáře, theList – seznam
    aktuálně získaných adresářů
void GetRegistrySubKeys(RegistryKey regKey, List<RegistryKey> theList)
{
    if (regKey != null)
    {
        RegistryKey rkey;
        if (regKey.SubKeyCount > 0)
        {
            theList.Add(regKey);
            for (int i = 0; i < regKey.SubKeyCount; i++)
            {
                try
                {
                    rkey = regKey.OpenSubKey(regKey.GetSubKeyNames()[i]);
                }
                catch { rkey = null; }
                GetRegistrySubKeys(rkey, theList);
            }
        }
        else
        {
            theList.Add(regKey);
        }
    }
}
```

Výpis 4: Rekurzivní procházení struktury klíče registru

Opět se může stát, že zadané klíče registrů nebudou existovat. Tyto objekty nenabývají žádných hodnot, tudíž postačí podmínka if(hodnota != null), při jejímž selhání opět hodnotu ignorujeme. Horší situace může nastat, pokud k danému registru nebudeme mít přístup. Na tuto anomálii neexistuje v jádru .NET efektivní metoda vracející true/false, jak to je u testování existence souborů nebo adresářů či vracení hodnoty null. Jediným řešením, aby NPW nevyhodil výjimku při odebírání skutečných hodnot registrů na základně hodnot z INI souboru, je metodu OpenSubKey objektu registru zapsat do bloku try/catch a vyhnout se tak vjímce SecurityException. Tento problém nemůžeme však dopředu eliminovat, proto při rekurzivním procházením např. klíče HKCU\Software může při tisícovkách klíčů a vlastností docházet ke zpomalení čtení registrů, protože se musí otestovat každý podklíč (v sekci porovnání rychlostí si tuto hypotézu otestujeme).

Důvod proč NPW vytváří soubor registrů, který je pak po vytvoření přidán k seznamu souborů zahrnutých do datového souboru je takový, že při obnově dat z datového souboru (popsáno níže) ho stačí jen rozbalit z datového souboru a spustit, a tím pak veškerá režie zásahu do registrů z pohledu programu odpadá, protože systém už si klíče a hodnoty zapíše do registrů sám.

K tomu aby tato operace proběhla úspěšně, je zapotřebí mít správný zápis hodnot v souboru registrů a správný zápis hlavičky souboru registrů (Windows Registry Editor Version 5.00). Jednoduše musí splňovat kritéria pro soubor registrů. Více informací o registrech najdete zde [11-14]. Klíče registrů se do souboru zapisují ve tvaru [absolutní cesta klíče] a pod něj jednotlivé hodnoty ve tvaru ”název hodnoty”=hodnota tak, jak ukazuje výpis 5. Hodnota může nabývat pěti různých typů hodnot:

- String
- Dword
- Binary
- MultiString
- ExpandString

Hodnotu registru ukládá NPW jako objekt nově vytvořené struktury RegistryValue, která má vlastnosti: název, hodnota a typ hodnoty. Hodnota je uložená jako objekt, protože, jak už bylo zmíněno, typů hodnot je více druhů. K rozlišení hodnoty jako takové pak slouží vlastnost typ hodnoty, který se nachází ve výčtovém typu RegistryValueKind. Při sestavování souborů registrů pak stačí zjistit jakého typu je zrovna zapisovaná hodnota a podle toho ji zapsat v takovém tvaru, v jakém je potřeba.

```
[HKEY_CURRENT_USER\Software\Microsoft\MediaPlayer\Setup]
" InstallResult "=dword:0
"UpdateTimeStamp"=hex:E,82,96,4B
"ColorPlayer"="#0063B0"
```

Výpis 5: Ukázka zápisu hodnot do souboru registrů

Hodnoty pod typem ExpandString dostaneme v ”přeložené” podobě, tzn. že například hodnota %USERPROFILE% se nám přemění na cestu k uživatelskému profilu. Jelikož si tuto hodnotu nemůžeme dovolit zapsat v ”přeloženém” tvaru do datového souboru, musíme při zápisu do souboru registrů zjistit, pod jakou proměnnou prostředí se tato cesta nachází. Všechny tyto proměnné prostředí dostaneme pomocí metody GetEnvironmentVariables třídy Enviroment. Tato metoda vrací slovník, kde klíčem je proměnná prostředí a hodnota je cesta nebo potřebný řetězec, který tento klíč reprezentuje. Jakmile se najde odpovídající klíč k již ”přeložené” hodnotě, použije se při zápisu do souboru registrů právě tento klíč namísto hodnoty.

Zápis samotný

Jakmile je kompletní seznam souborů (včetně souboru registrů) k zápisu, všechny tyto soubory NPW ukládá do datového souboru bez jakékoliv komprimace. Jak pacovat se soubory poslouží informace na [7,15,18]. K vytvoření tohoto souboru slouží třída DataFile.cs a její metoda WriteDataFile, která má tyto parametry: objekt FileStream obsahující absolutní cestu souboru, název absolutní cesty souboru a počítadlo. V cyklu, který projíždí vytvořený seznam souborů, se zapisují jednotlivé soubory do datového souboru postupně. Počítadlo slouží k rozlišení, zda-li se jedná o první soubor, protože první soubor nám označuje vytvoření datového souboru, tudíž FilleAcces jako parametr objektu FileStream se použije Write, pro ostatní soubory to bude Append, tzn. že těmito soubory bude datový soubor už jen narůstat.

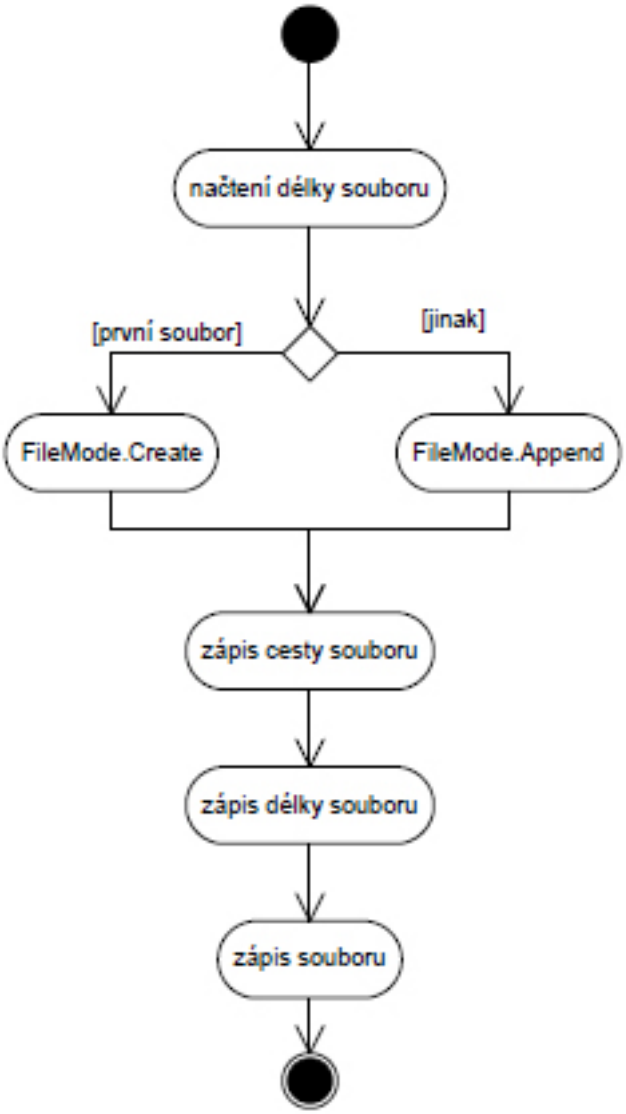
Důležitým prvkem je složení datového souboru. Nejde jenom o zápis souboru, ale zároveň je třeba připravit půdu pro případnou obnovu dat. Z toho důvodu je zapotřebí v datovém souboru uchovávat i jiné informace, než jenom samotné soubory. Před zápisem souboru se zapíše cesta souboru a jeho velikost, protože při čtení binárního souboru postupujeme stejně jako při zápisu (od začátku do konce) tak, aby NPW byl schopný přečíst kam má soubor obnovit a jaká délka je mu v datovém souboru vyhrazena. Aktivitní diagram na obrázku 5 zobrazuje zjednodušeně proces zápisu souborů do datového souboru.

Při zápisu souborů může nastat jistý konflikt. Pokud soubor, který je třeba zapsat do datového souboru, používá jiný proces, nastane výjimka UnauthorizedAccesException. Netýká se to až tak typu souboru, ale jde převážně o proces, který tento soubor používá. Pokud tedy daný soubor používá proces, který nepovolí čtení souboru jiným procesem během jeho používání, vyhodí se již zmíněná výjimka. Bohužel nelze nijak tuto skutečnost zjistit předem, proto pro stabilitu procesu zápisu dat do datového souboru musíme tuto výjimku odchytit pomocí bloků try/catch a tento soubor se jednoduše do datového souboru nepoužije.

4.2.4 Obnova dat z datového souboru

Příprava pro obnovu

Proces obnovy dat je téměř totožný se zápisem dat do datového souboru, respektive s podobností funkcí. Data, která se berou z INI souboru, slouží k tomu, abychom smazali



Obrázek 5: Aktivitní diagram zápisu souborů do datového souboru

soubory a registry před jejich obnovou, tzn. že budou aktuální k datu vytvoření datového souboru.

Vlastnosti, které NPW zpracovává k vytvoření seznamu souborů jsou:

- FolderTreesToRemoveToResetToDefaults
- IndividualFilesToRemoveToResetToDefaults
- ExcludeFilesToRemoveToResetToDefaults

Wildcard masky se smí používat už i u IndividualFilesToRemoveToResetToDefaults a to u názvu souboru a u ExcludeFilesToRemoveToResetToDefaults pouze jako první znak názvu souboru. Princip je stejný jako u vytváření seznamu souborů pro zápis do datového souboru. Nejdříve se načtou všechny soubory (rekurzivně a pomocí metody GetFiles) a po té se z nich vyjmou konkrétní soubory.

Tento postup platí také pro sekci registrů, konkrétně se jedná o vlastnosti:

- RegistryTreesToRemoveToResetToDefaults
- IndividualRegistryValuesToRemoveToResetToDefault
- RegistryTreesToExcludeToResetToDefaults
- RegistryKeysToExcludeToResetToDefaults
- RegistryValuesToExcludeToResetToDefaults

Odstranění souborů a registrů

Nyní, když má NPW sestavený seznam souborů a registrů, smaže tato data a může začít s obnovou dat z datového souboru. K odstranění souborů si vystačíme s metodou Delete třídy File. Pro odstranění registrů je zapotřebí sofistikovanější metody. I když v konfiguračním souboru jsou vlastnosti, které určují smazat celé klíče registru, NPW odstraňuje pouze hodnoty klíče. Vychází se z faktu, že klíče bez hodnot se dají chápat jako by tam vůbec nebyly. Toto řešení je z toho důvodu, že nejsou takové metody, které by plně posloužily potřebám konfiguračního souboru, a proto vytvářet vlastní řešení je víceméně zbytečné.

Důležitým faktem při použití metody OpenSubKey pro otevření klíče registru, ze kterého bude NPW odstraňovat hodnoty, je přidat k paramterům metody parametr "writable" a nastavit na "true", defaultní hodnota je totiž "false", jinak při pokusu odstranit z tohoto klíče nějakou hodnotu nastane výjimka UnauthorizedAccesException.

Obnova samotná

Předpokládá se, že NPW čte data z datového souboru v takovém pořadí, jako je zapsal, tzn. od začátku až dokonce. Jelikož známe strukturu datového souboru, tak NPW

postupně přečte cestu souboru, vytvoří cestu pro tento soubor pokud tato cesta neexistuje, přečte velikost souboru a podle velikosti přečte daný soubor. Následně pak tento soubor zapíše na disk přesně tam, odkud ho před zápisem do datového souboru kopíroval. Nakonec se spustí soubor registrů a systém obnoví zapsané registry zpět do sekce registrů. Více o spouštění spustitelných souborů na [7,17]. Aktivitní diagram na obrázku 6 zobrazuje zjednodušeně proces obnovy souborů z datového souboru.

Při obnově souborů může opět nastat jistý konflikt. Pokud soubor, který je třeba obnovit nebo smazat před obnovou, používá jiný proces, nastane výjimka UnauthorizedAccesException. Netýká se to až tak typu souboru, ale jde převážně o proces, který tento soubor používá. Pokud tedy daný soubor používá proces, který nepovolí čtení souboru jiným procesem během jeho používání, vyhodí se již zmíněná výjimka. Bohužel nelze nijak tuto skutečnost zjistit předem, proto pro stabilitu procesu obnovy dat musíme tuto výjimku odchytit pomocí bloků try/catch. I po tomto konfliktu je třeba zachovat čtení datového souboru, proto bude pokračovat dále, s tím, že podle uložené délky tohoto souboru přečte délku a bude pokračovat ve čtení datového souboru, aniž by se NPW dále pokoušel o zápis tohoto souboru na disk.

Nesmí se zapomenout na samotný datový soubor. Pokud nebude v této fázi použití NPW existovat, hrozí výjimky FileNotFoundException. Tento konflikt lze však elegantně pořešit klasickou metodou Exists třídy File.

4.2.5 Konfigurační soubor aplikace

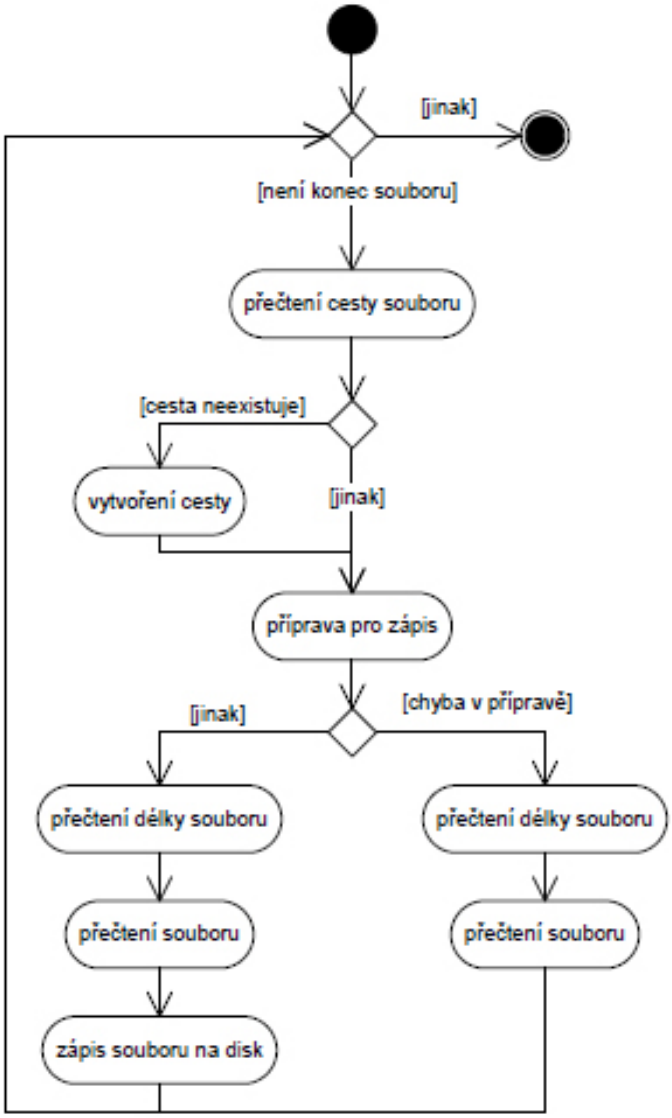
Pro stanovení cest k různým souborům a registrům existuje jejich zápis v konfiguračním souboru aplikace. Tím se ulehčuje správa nad těmito hodnotami. Podoba konfiguračního souboru je uvedená na výpisu 6.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key="ini_file " value="OPW10ADM.INI" />
    <add key="HKCU" value="Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\Shell␣
        Folders" />
    <add key="officeHKCU10" value="Software\\Microsoft\\Office\\10.0\\Common\\General" />
    <add key="officeHKCU11" value="Software\\Microsoft\\Office\\11.0\\Common\\General" />
    <add key="officeHKCU12" value="Software\\Microsoft\\Office\\12.0\\Common\\General" />
    <add key="dataFile" value="./DataFile.DAT" />
    <add key="errorFile" value="./errors.txt " />
    <add key="regFile" value="TEMP\\regFile.reg" />
  </appSettings>
</configuration>
```

Výpis 6: Konfigurační soubor aplikace New Profile Wizard

Význam jednotlivých klíčů a jejich hodnot:

- ini_file - cesta a název ke konfiguračnímu souboru
- HKCU - cesta ke klíči registru se seznamem souborových tokenů



Obrázek 6: Aktivitní diagram obnovy souborů z datového souboru

- officeHKCU10 - cesta ke klíči registru se seznamem subsouborových tokenů pro MS Office verzi 10.0
- officeHKCU11 - cesta ke klíči registru se seznamem subsouborových tokenů pro MS Office verzi 11.0
- officeHKCU12 - cesta ke klíči registru se seznamem subsouborových tokenů pro MS Office verzi 12.0
- dataFile - cesta a název k datovému souboru
- errorFile - cesta a název k souboru s hlášením o chybách
- regFile - cesta a název k souboru registrů

4.2.6 Třídní diagram aplikace

Seznam tříd a jejich rozvržení v aplikaci znázorňuje třídní diagram na obrázku 7.

4.2.7 Uživatelské rohraní NPW

Rozhodně se NPW nesnaží kopírovat uživateské rozhraní PW, jde převážně o funkcionalitu, proto UI (obrázek 8) obsahuje dvě základní tlačítka pro zápis do datového souboru a pro obnovu dat z datového souboru včetně ResetToDefaults funkce. Dále se zobrazuje, na čem momentálně NPW pracuje.

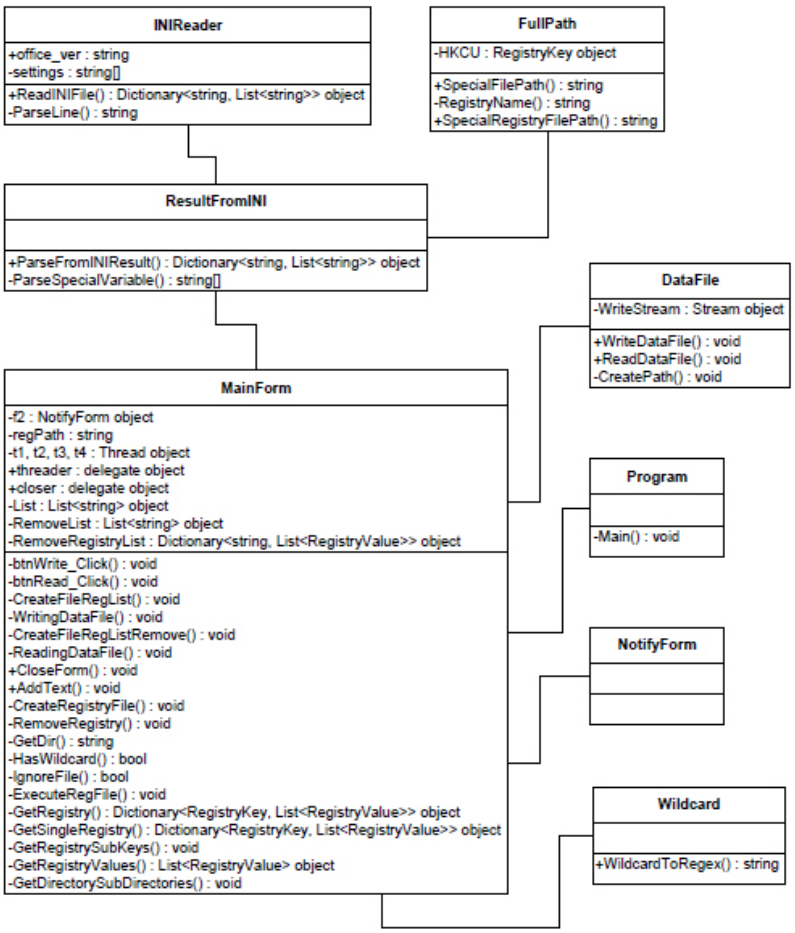
Při zápisu datového souboru dochází k sestavování absolutních cest k souborům a vytváření seznamu registrů pro zápis do souboru registrů. Tuto operaci prezentuje notifikační okno Info, s textem ”Creating file and registry list ...”. Po sestavení tohoto seznamu začne samotný zápis datového souboru. Soubory, které jsou čteny a následně zapisovány, se zobrazují vespod formuláře. Takhle máme přehled co NPW dělá a jak dlouho mu tyto operace trvají.

Při obnově dat se taktéž zobrazuje (při sestavování cest k souborům a vytváření seznamu registrů) notifikační okno Info, ale při obnově samotné už jen zpráva o probíhající operaci ”Restoring files and registries ...”.

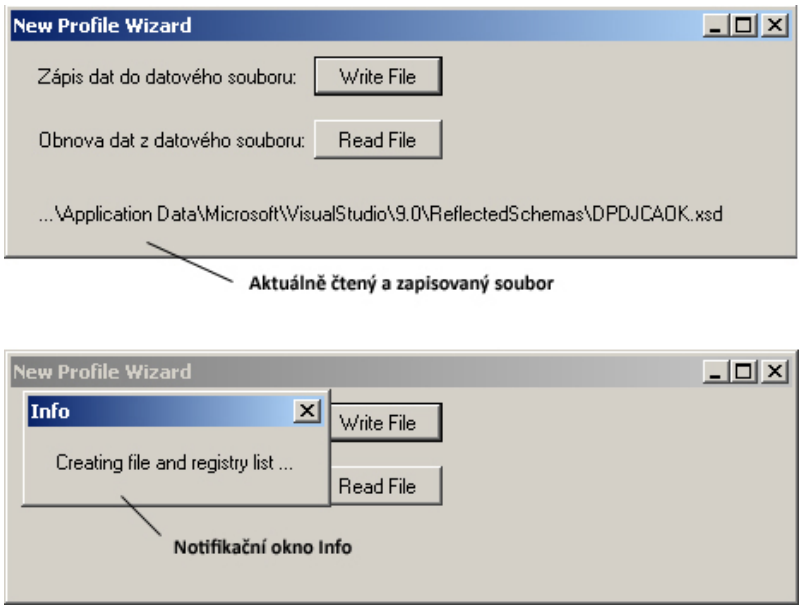
Každá správně dokončená operace je potvrzena zprávou ”Data file created.” či ”Files restored.”. V případě neexistujícího datového souboru zpráva ”Data file does not exist.”.

4.3 Porovnání NPW a PW

K porovnání těchto dvou programů je zapotřebí, aby vytvářely datový soubor, či obnovovaly z něj soubory, které nejsou používány žádným jiným programem, a aby uvedené cesty v konfiguračním souboru opravdu existovaly. Již ve výše zmíněném postupu, jakým NPW kontroluje tyto anomálie, bylo zapotřebí se těchto anomálií zbavit, nebo nejlépe je ignorovat. To, jakým způsobem tyto anomálie řeší PW od Microsoftu, je řešením, kterým se NPW nevydává. NPW tyto anomálie automaticky ignoruje a pokračuje v práci, aniž by nějak dal najevo uživateli jaké anomálie při zápisu



Obrázek 7: Třídní diagram aplikace New Profile Wizard



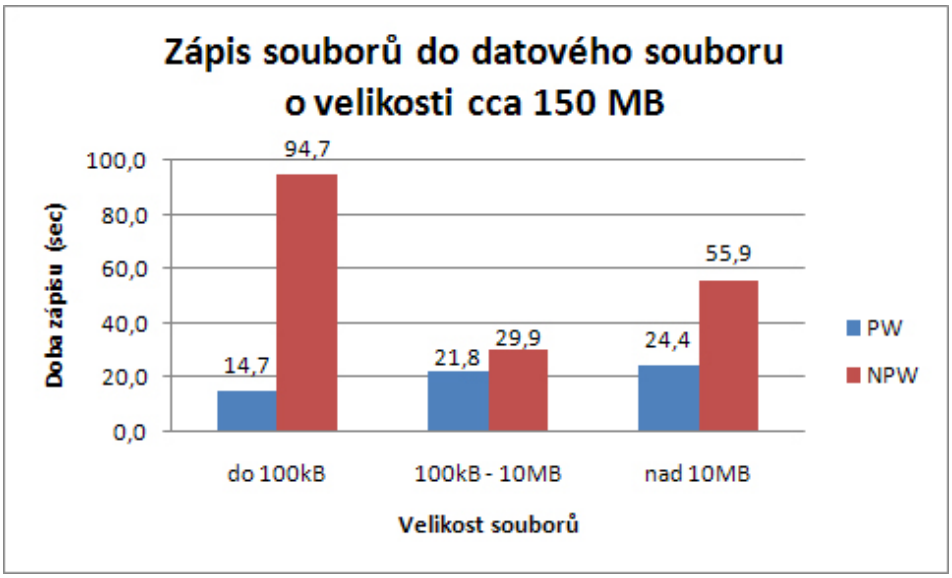
Obrázek 8: Uživatelské rohraní NPW

nebo obnově nastaly. PW tohle řeší tím, že uživateli zobrazí zprávu o chybě v parsování konfiguračního souboru, nebo při nemožnosti přístupu k souborům, které jsou používané jinými programy. Důvod, proč tohle NPW nerealizuje, je takový, že potřebujeme výsledek ať už v jakékoliv podobě a případné chyby o neexistujících adresářích a dalších anomáliích ukládá do pomocného souboru errors.txt. V konečné fázi je to irelevantní, a toto řešení jsem nechal čistě na mně.

Testování probíhalo v několika kombinacích celkového počtu souborů a registrů, velikostí souborů a s použitím bloků try/catch či nikoliv. Je totiž dost možné, že odchytávání výjimek může mít za následek pomalejšího vytváření datového souboru, či obnovy z něj. Dále je možné, že ukládání délky souboru a názvu souboru před samotným souborem povede ke zvětšení datového souboru a ke dvěma operacím navíc, které ve větším množství souborů může opět ubrat na výkonosti programu. V konečné podobě NPW budou tyto zatím jen hypotetické zpomalovací mechanizmy použity, protože jsou zapotřebí k bezproblémovému běhu NPW při vytváření datového souboru a obnovy z něj. Těmito testy si také ověříme, zda-li nám tyto mechanizmy nezpomalují práci NPW na úkor jiných možných řešení, které nebyly použity (ať už to bylo možné v rámci použité platformy .NET Framework či nikoliv).

4.3.1 Testování rychlostí zápisu souborů

Jako první testování, které bylo prováděno, je testování kompletní aplikace NPW s aplikací MS Office Profile Wizard. Aplikace NPW obsahuje veškerá ošetření a datový soubor obsahuje před každým zápisem souboru jeho délku a cestu k souboru. Pro všechny testy bylo zapotřebí provést několik měření doby trvání práce těchto programů, abychom do-



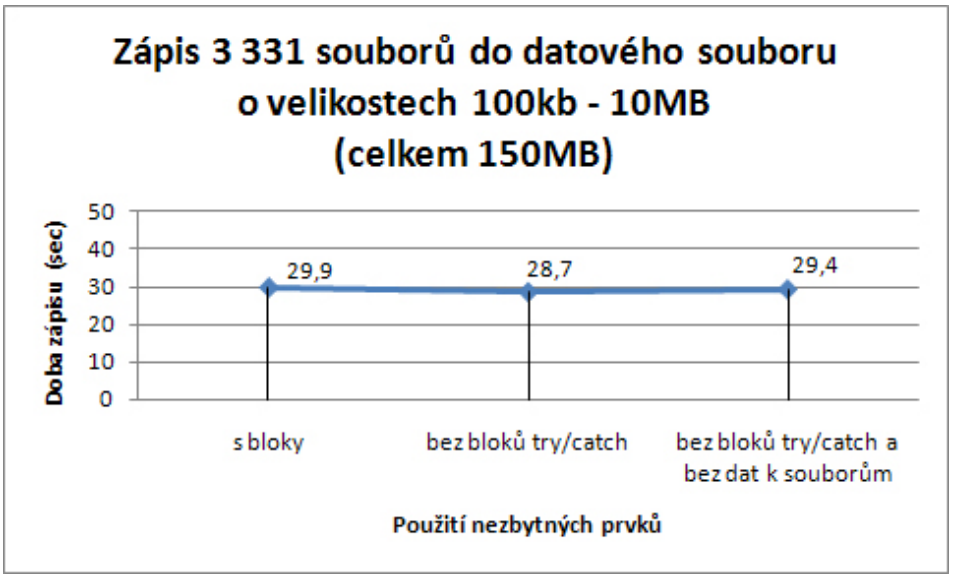
Obrázek 9: Graf rychlostí zápisu souborů do datového souboru

stali věrohodné informace, tzn. odstranila se odlehlá pozorování a udělal se průměr z ostatních hodnot.

Testování zápisu se dělí na tři složky, a to podle velikosti souborů a celkového počtu těchto souborů, avšak celková velikost těchto složek je cca 150MB. První složkou jsou soubory, které mají velikost do 100kB a jejich počet je řádově v desítkách tisíc. Druhou složkou jsou soubory, které mají velikost od 100kB do 10MB a jejich počet je řádově v tisících. Poslední složkou jsou soubory, které mají velikost větší než 10MB a jejich počet je velmi malý. Pro srovnání rychlostí by eventuelně stačily velikosti souborů uvedené ve druhé složce, protože tyto velikosti a počty odpovídají běžné práci uživatele. Ostatní uvedené složky slouží k zatížení programů a výsledek nám určí slabiny těchto programů.

Graf na obrázku 9 zobrazuje rozdíl rychlosti práce obou aplikací. Je vidět, že enormní množství souborů do 100kB pro zápis vede ke značnému zpomalení aplikace NPW oproti PW. Co se týče souborů, které by odpovídaly běžné práci uživatele (100kB - 10MB), byla rychlost zápisu dosti podobná, ale i přesto PW dokázal tyto soubory zpracovat a zapsat o něco rychleji. U poslední složky dokázal PW pracovat s dvojnásobnou rychlostí.

Z dříve zmiňovaných pochybností s použitím bloků try/catch a zápisu délky a názvu cesty do datového souboru před každý zapisovaný soubor, bylo provedeno testování, zda-li tyto mechanizmy nezpomalují práci aplikace NPW. Testování bylo provedeno na druhé složce předchozího testu, kde velikosti souborů se pohybovaly v rozmezí od 100kb do 10MB a jejich počet byl 3331. Graf na obrázku 10 zobrazuje hodnotu s použitím bloků try/catch a s informacemi souboru, hodnotu bez použití bloků try/catch a hodnotu bez použití veškerých mechanizmů. Domněnku, že by tyto mechanizmy zpomalovaly práci aplikace NPW, můžeme na základě tohoto grafu zamítnout, a můžeme konstatovat, že



Obrázek 10: Graf rychlostí zápisu souborů s a bez použití důležitých prvků

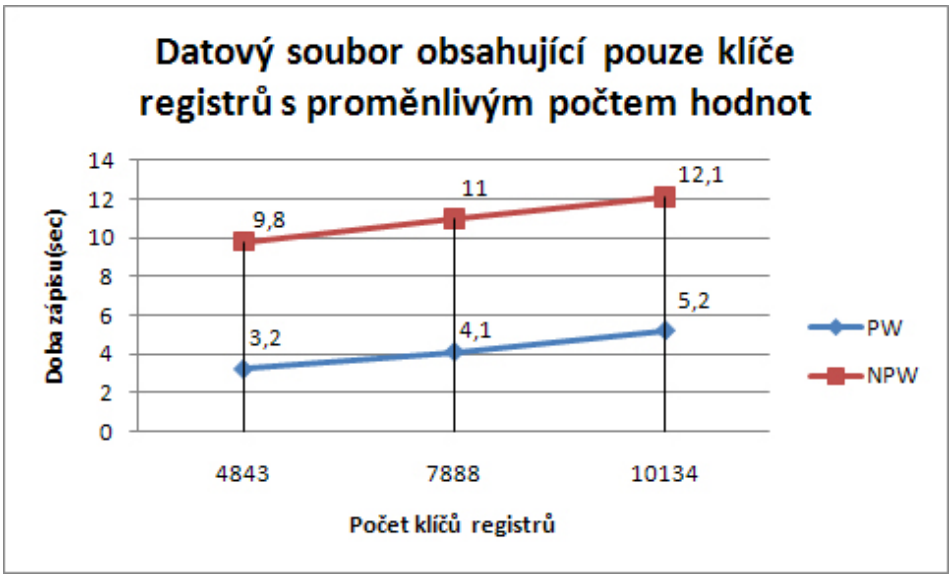
použití ochranných prvků a pomocných prvků pro obnovu z datového souboru nijak neovlivňuje práci aplikace NPW.

4.3.2 Testování rychlostí zápisu registrů

K tomuto testování se vytvářel datový soubor pouze z klíčů registrů a jejich hodnot. Jelikož způsob jakým PW zapisuje klíče registrů není známý, cílem tohoto testování bylo zjistit, zda způsob, kterým tuto operaci realizuje NPW, je dostatečně efektivní. Tudíž vytvořit soubor registrů, kde klíče registrů se získávaly rekurzivně z registrů systému. Graf na obrázku 11 jasně ukazuje, že aplikace PW je rychlejší. Bylo však použito velké množství klíčů, protože jedině takhle se dá porovnat způsob práce těchto programů. Je totiž samozřejmé, že při malém počtu klíčů, řádově ve stovkách, nebudou rozdíly na první pohled patrné.

4.3.3 Testování rychlostí obnovy dat

K testování rychlostí obnovy dat z datového souboru bylo použito stejných složek a vytvořeného datového souboru jako při testování rychlostí jeho vytvoření. Pro připomenutí: testování obnovy se dělí na tři složky, a to podle velikosti souborů a celkového počtu těchto souborů, avšak celková velikost těchto složek je cca 150MB. První složkou jsou soubory, které mají velikost do 100kB a jejich počet je řádově v desítkách tisíc. Druhou složkou jsou soubory, které mají velikost od 100kB do 10MB a jejich počet je řádově v tisících. Poslední složkou jsou soubory, které mají velikost větší než 10MB a jejich počet je velmi malý. Pro srovnání rychlostí by eventuelně stačily velikosti souborů uvedené



Obrázek 11: Graf rychlostí zápisu klíčů registrů a jejich hodnot

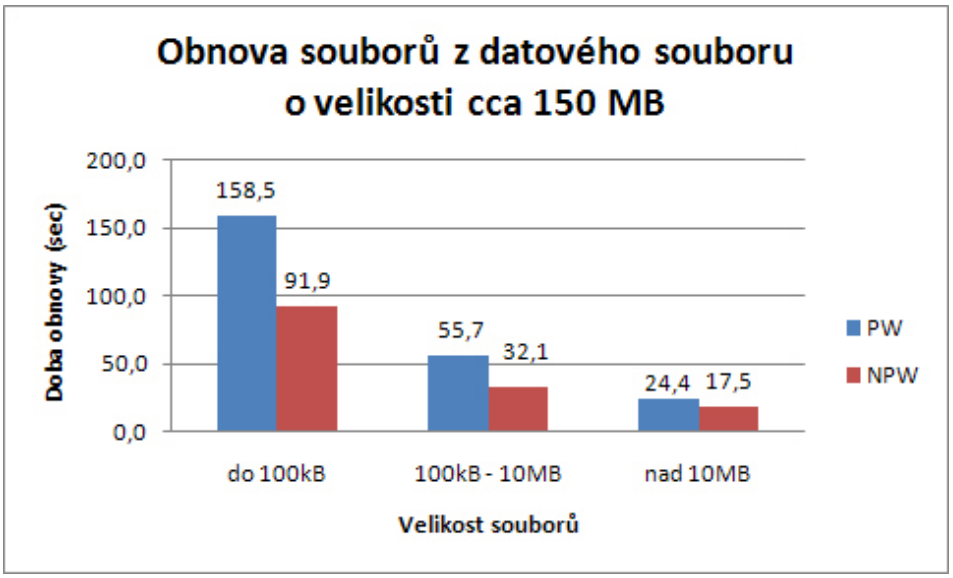
ve druhé složce, protože tyto velikosti a počty odpovídají běžné práci uživatele, ostatní uvedené složky slouží k zatížení programů a výsledek nám určí slabiny těchto programů.

Graf na obrázku 12 ukazuje ve všech případech, že rychlost s jakou NPW obnovuje soubory z datového souboru je nižší, tudíž práce NPW je efektivnější při obnově. S menším počtem souborů se rozdíly zmenšují. Důležitým faktorem je tedy opět druhá složka - práce aplikace NPW je téměř o 100% efektivnější, což je velmi pozitivní výsledek.

4.3.4 Testování rychlostí obnovy registrů

K porovnání rychlostí, s jakou aplikace obnovují registry, bylo vytvořeno 10000 klíčů registrů různě v sobě zanořených, kde každý registr měl desítky hodnot, abychom nasimulovali co největší zátěž programů. Jelikož není znám způsob, jakým tuto operaci realizuje PW, je možné alespoň určit zda způsob, kterým NPW obnovuje registry, je oproti PW efektivní či nikoliv. Pro připomenutí: NPW realizuje operaci s registry pomocí souboru registrů a veškerou obnovu nechává na systému.

K tomuto testování není třeba grafu k vizualizaci a představě, protože výsledky jsou enormně rozdílné. Je samozřejmé, že při malém počtu klíču registrů či hodnot nebude rozdíl znát, ale při testování již zmiňovaných 10000 registrů byl způsob obnovy NPW značně rychlejší. Z uživatelského rozhraní PW lze odvodit, že PW pracuje s každým klíčem registru či hodnotou zvlášť, protože při procesu obnovy zobrazuje právě zpracovávaný registr, což při spuštění souboru registru nelze. Tedy nechat obnovu dat na systému je mnohonásobně rychlejší.



Obrázek 12: Graf rychlostí obnovy souborů z datového souboru

4.4 Shrnutí aplikace NPW

Je zřejmé, že různé způsoby implementace problémů vedou k různým výkonostním výsledkům. Implementace NPW není výjimkou. Záleží už jen na tom, jak moc dobře byla zvolena konkrétní implementace. Z výsledků testování lze konstatovat, že implementace zápisu souborů hraje důležitou roli, ale s pozitivním výsledkem, kde při zápisu souborů pomocí NPW, kde jejich počet a velikost je srovnatelná s běžnou prací uživatele, byla rychlost srovnatelná s aplikací MS Office Profile wizard.

Rychlost obnovy souborů, kde jejich počet a velikost je srovnatelná s běžnou prací uživatele, byla větší o 100% oproti aplikaci PW. Je ale možné, že PW bude dbát větší pozornosti zápisu souborů na disk pomocí různých mechanizmů, např. kontrolních součtů apod., ale stále je toto jenom domněnkou, protože není znám způsob implementace těchto operací pomocí PW.

Zápis registrů pomocí NPW nedopadl ve srovnání s PW vůbec tragicky, ale důležitým faktem je, že NPW vytváří soubor registrů, což ovlivňuje obnovu registrů. Z provedených testů lze tedy tvrdit, že způsob, jakým NPW pracuje s registry, je mnohem efektivnější. Přestože zápis registrů trval déle, jejich obnova zabrala značně méně času.

5 Uživatelské rozhraní pro MS Office Profile Wizard

Tato aplikace je napsaná na platformě .NET Framework 2.0 jazykem C# 2.0 a je vytvořena jako Windows Forms Aplikace.

Aplikace uživatelského rozhraní používá stejnou třídu pro parsování souboru, tak jako NPW. Objekt již definovaného typu Dictionary<string, List<string>> není nadále upravován a používají se jeho hodnoty přímo. Je to z toho důvodu, že uživateli nepotřebujeme reprezentovat absolutní cesty k souborům či adresářům, neboť tyto hodnoty nejsou pro něj v této fázi podstatné.

5.1 Princip aplikace uživatelského rozhraní

Jelikož tato aplikace slouží pro práci s konfiguračním souborem pro MS Office Profile Wizard, jediný datový zdroj bude právě tento soubor, nad kterým budou všechny změny řízeny a ukládány. Podstatou je tedy načíst tento soubor a prezentovat hodnoty jednotlivých vlastností přímo na obrazovku uživatele. Tyto hodnoty budou uloženy ve výše zmíněném objektu, který je pro každé načtení konfiguračmího souboru v rámci aplikace unikátní instancí třídy MainObject.cs. Veškeré prováděné změny se provádí do tohoto objektu (představme si ho jako konfigurační soubor, ale ve virtuální podobě) a následně při pokynu uživatele uložit tyto změny, přepíše se celý konfigurační soubor tímto objektem a vzniká nám upravený konfigurační soubor.

5.1.1 Načtení konfiguračního souboru

Pro vytvoření nového konfiguračního souboru, nebo načtení již existujícího konfiguračního souboru, slouží menu aplikce. Pro nový File -> New, pro načtení File -> Open, popřípadě použití tlačítek na ovládacím panelu aplikace.

Při vytváření nového konfiguračního souboru je zapotřebí vytvořit nový objekt Dictionary<string, List<string>>, protože se nezačíná s parsováním již existujícího souboru. Ze seznamu všech vlastností, které jsou pro MS Office Profile Wizard povoleny, se tento objekt vytvoří s tím, že místo pro hodnoty pod jednotlivými klíči zůstane prázdné a postupně se bude zaplňovat v závislosti na práci uživatele s programem.

Při načítání již existujícího konfiguračního souboru si uživatel vybere soubor z disku a tento soubor se poté začne parsovat třídou INIReader.cs. Výstupem bude objekt Dictionary<string, List<string>> s načtenými vlastnostmi a jejich hodnotami. Může se stát, že konfigurační soubor neobsahuje všechny vlastnosti, proto se vytvořený objekt doplní chybějícími vlastnostmi ze seznamu vlastností.

5.1.2 Ukládání změn do souboru

Aplikace obsahuje vlastnost Changes, která indikuje zda byly provedeny změny v hodnotách vlastností. Při každé provedené změně se tato vlastnost nastaví na True, tím dá aplikace pro ostatní prvky najevo, že byly provedeny změny, a že je možné tyto změny uložit do konfiguračního souboru.

Provedené změny se projeví v objektu MainObject ihned po překliknutí na jinou vlastnost, nebo po uložení změn do souboru. Při překliknutí na jinou vlastnost se provedou změny pouze v objektu MainObject, nikoliv v souboru. Změny v souboru se provedou po kliknutí na položku File -> Save v menu aplikace, nebo pomocí tlačítka na ovládacím panelu. Pokud aplikaci zavíráme a existují změny, které do souboru nebyly uloženy, aplikace zobrazí notifikační okno o této skutečnosti a nabídne změny uložit.

Alternativou pro uložení konfiguračního souboru je použití funkce "Uložit jako" pomocí položky v menu File -> Save As ... se vytvoří nový konfigurační soubor a tento soubor se použije jako aktuálně používaný konfigurační soubor objektu MainObject.

5.2 Seznam vlastností a jejich hodnot

Jelikož pro MS Office Profile Wizard (PW), je důležitý přesný počet vlastností, existuje v aplikaci výčtový typ se všemi těmito vlastnotmi a nemůže se stát, aby konfigurační soubor obsahoval nepovolené vlastnosti. Toto statické řešení je bohatě postačující a můžeme na tom postavit myšlenku aplikace. Seznam těchto vlastností je reprezentován komponentou TreeView jako uzly této komponenty. Při projíždění těchto uzlů se zobrazují hodnoty ve vedlejší komponentě ListView, ostatně jak je vidno na obrázku 13, který reprezentuje náhled na celou aplikaci.

Seznam všech vlastností a jejich princip:

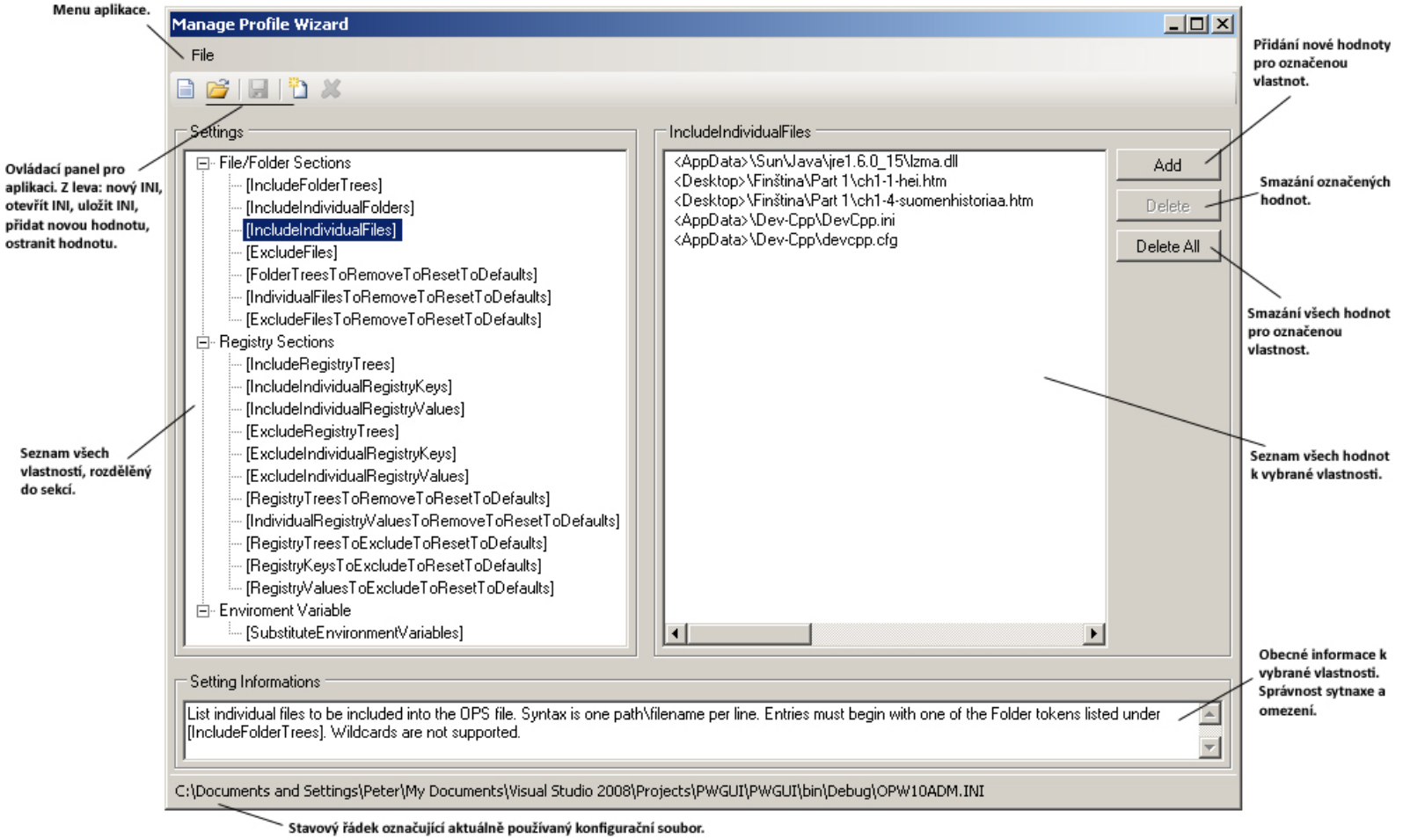
- IncludeFolderTrees - seznam adresářů, které budou zahrnuty v datovém souboru (zahrnuje všechny poadresáře)
- IncludeIndividualFolders - seznam adresářů, které budou zahrnuty v datovém souboru
- IncludeIndividualFiles - seznam souborů, které budou zahrnuty v datovém souboru
- ExcludeFiles - seznam souborů, které nebudou zahrnuty v datovém souboru
- FolderTreesToRemoveToResetToDefaults - seznam adresářů, které budou smazány před obnovou (zahrnuje všechny poadresáře)
- IndividualFilesToRemoveToResetToDefaults - seznam souborů, které budou smazány před obnovou
- ExcludeFilesToRemoveToResetToDefaults - seznam souborů, které nebudou smazány před obnovou
- SubstituteEnvironmentVariables - seznam proměnných prostředí, které budou nahrazeny v hodnotách registru typu REG_EXPAND_SZ
- IncludeRegistryTrees - seznam klíčů registrů, které budou zahrnuty v datovém souboru (zahrnuje všechny podklíče)

- IncludeIndividualRegistryKeys - seznam klíčů registrů, které budou zahrnuty v datovém souboru
- IncludeIndividualRegistryValues - seznam hodnot klíčů registrů, které budou zahrnuty v datovém souboru
- ExcludeRegistryTrees - seznam klíčů registrů, které nebudou zahrnuty v datovém souboru (zahrnuje všechny podklíče)
- ExcludeIndividualRegistryKeys - seznam klíčů registrů, které nebudou zahrnuty v datovém souboru
- ExcludeIndividualRegistryValues - seznam hodnot klíčů registrů, které nebudou zahrnuty v datovém souboru
- RegistryTreesToRemoveToResetToDefaults - seznam klíčů registrů, které budou smazány před obnovou (zahrnuje všechny podklíče)
- IndividualRegistryValuesToRemoveToResetToDefaults - seznam hodnot klíčů registrů, které budou smazány před obnovou
- RegistryTreesToExcludeToResetToDefaults - seznam klíčů registrů, které nebudou smazány před obnovou (zahrnuje všechny podklíče)
- RegistryKeysToExcludeToResetToDefaults - seznam klíčů registrů, které nebudou smazány před obnovou
- RegistryValuesToExcludeToResetToDefaults - seznam hodnot klíčů registrů, které nebudou smazány před obnovou

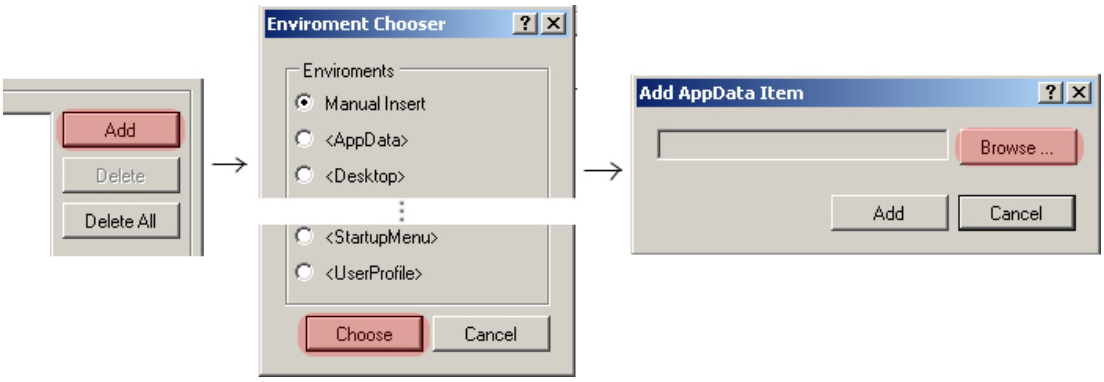
Pro přidávání hodnot ke všem vlastnostem, s výjimkou SubstituteEnvironmentVariables, slouží třída AddItem.cs, kde pomocí přepínače Switch rozlišujeme, jaký formulář pro přidání hodnoty použijeme. Vstupem pro rozhodování jsou tyto faktory: zda-li se jedná o adresář, soubor, klíč registru nebo určitá hodnota klíče registru a faktory: speciální cesta k adresáři či souboru a kořenový klíč registru. Po kliknutí na jeden ze tří prvků aplikace: tlačítko Add, tlačítko Add z ovládacího panelu aplikace a položka Add v menu komponenty ListView, se zobrazí formulář, který slouží k upřesnění těchto kritérií. Po upřesnění těchto kritérií se zobrazí samotný formulář pro přidání hodnoty.

Zadání hodnoty je řízeno tak, aby uživatel vložil hodnoty, které v registrech či na disku systému opravdu existují. Výše zmiňovaný přepínač, tedy na základě aktivní vlastnosti a kritérií, zobrazí příslušný formulář pro přidání hodnoty, a to po kliknutí na tlačítko Browse formuláře Add ”konkrétní” Item. Tento proces je zobrazen na obrázku 14.

Každý tento formulář pro přidání hodnoty se od sebe liší, protože se vymezuje na přidání odlišných hodnot. Tyto formuláře a jejich práce jsou popsány níže. Po výběru hodnoty z těchto formulářů se tato hodnota zobrazí ve formuláři Add ”konkrétní” Item v podobě, se kterou pak bude vložena do konfiguračního souboru.



Obrázek 13: Uživatelské rozhraní pro MS Office Profile Wizard



Obrázek 14: Proces přidání hodnoty pro aktivní vlastnost

Způsob, jakým formuláře mezi sebou komunikují, je následující. Pro zobrazení formuláře potřebujeme potřebné parametry (kritéria), které třídy těchto formulářů zpracovávají. Tyto parametry se zadávají při zakládání nového objektu formuláře, jelikož konstruktor třídy formuláře tyto parametry vyžaduje. Každá třída formuláře má definovanou událost, která se u zaregistrovaných posluchačů zachytí když nastane. Posluchači dostanou rozšířený objekt třídy EventArgs obsahující data, která požadují. Celý tento proces funguje napříč celé aplikace a je základním kamenem komunikace mezi objekty formulářů. Třída formuláře dostane parametry, na jejichž základě zobrazí či povolí uživateli konkrétní formulář, kde po výběru či zadání hodnot vrátí zpět třídě, která tento objekt vytvářela, vybranou či zadanou hodnotu. Tento postup je demonstrován na výpisu 7.

```
// definice delegáta, jehož parametry jsou objekt, který vyvolává událost, a rozšířený EventArgs
    objekt
public delegate void FormData(object sender, AddItemEventArgs e);
// událost typu delegáta
public event FormData OnAddClick;

// vyvolání události, kterou zachytí všichni registrovaní posluchači
// objekt, který vyvolává tuto událost je skryt pod klíčovým slovem this, který je ukazatelem na
    aktuálnání instanci objektu
// parametry, které budou posluchači odebírat se předávají do nové instance třídy
    AddItemEventArgs
OnAddClick(this, new AddItemEventArgs(textBox.Text));

// třída AddItemEventArgs dědí z třídy EventArgs a je rozšířena právě o vlastnosti parametrů,
    které je potřeba předat posluchačům
public class AddItemEventArgs : EventArgs
{
    protected string item;
    public string Item
    {
        get { return item; }
        set { item = value; }
```

```
    }

    // parametr item pro posluchače
    public AddItemEventArgs(string item)
    {
        this.item = item;
    }
}

// vytvoření instance třídy AddItem
AddItem aiform = new AddItem(itemType, settingType, Setting.SettingName);
// registrování posluchače pro událost OnAddClick
aiform.OnAddClick += new AddItem.FormData(OnAddClick);

// jakmile nastane událost, posluchač ji zachytí, spustí metodu OnAddClick a vezme si potřebný
    parametr
void OnAddClick(object sender, AddItemEventArgs e)
{
    string item = e.Item;
}
```

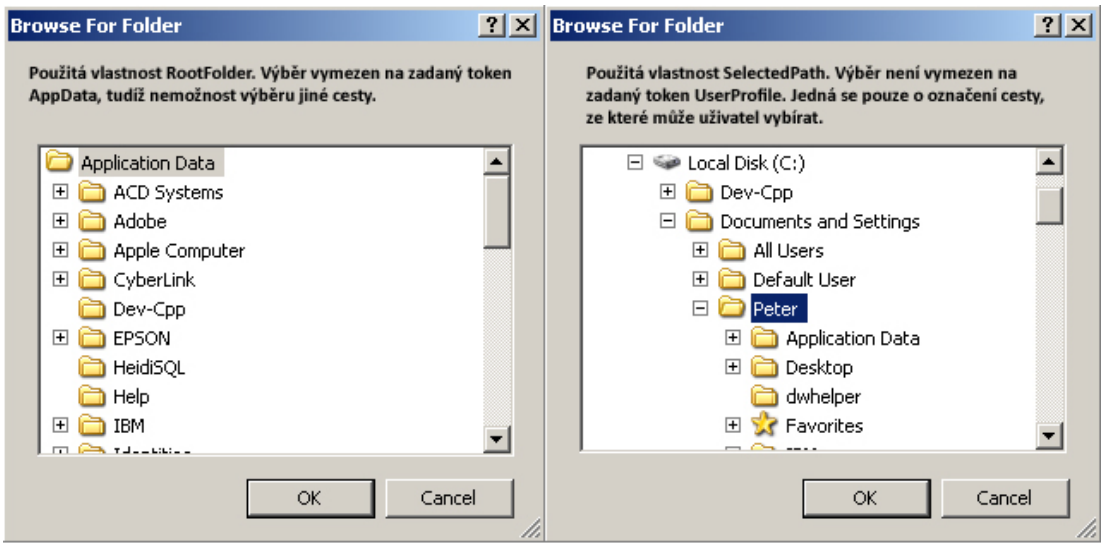
Výpis 7: Proces komunikace formulářů

5.2.1 Adresáře

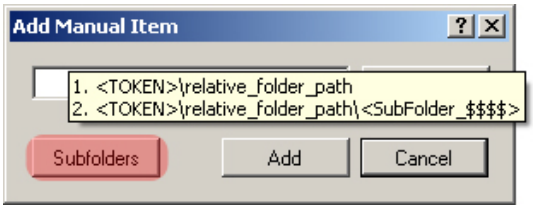
Pro přidávání adresářů slouží komponenta FolderBrowserDialog. Jako parametr pro zobrazení speciální cesty slouží předem vybraný token ze seznamu kritérii. FolderBrowserDialog má totiž vlastnost RootFolder, která nabývá hodnot z výčtového typu Environment.SpecialFolder a proto výběr adresářů se nám omezí pouze na zadaný token z kritérií. Výčtový typ Environment.SpecialFolder však neobsahuje všechny tokeny, které mohou být v konfiguračním souboru použity. K řešení tohoto problému poslouží vlastnost SelectedPath, díky které můžeme zobrazit výběr adresářu právě na vybrané cestě. Jediným problémem je, že uživatel může například překlikávat o hiearchii výše, a tudíž dostat špatnou cestu vzhledem k vybranému tokenu z kritérií. Proto existuje ošetření této anomálie a uživateli se zobrazí hlášení o neplatné cestě, a tedy výběr cesty musí opakovat. Obrázek 15 ukazuje výběr zleva pomocí vlastnosti RootFolder a pomocí SelectedPath.

5.2.2 Soubory

Pro přidávání souborů slouží komponenta OpenFileDialog. Opět parametr vymezující cestu k souborům slouží pro zadání vlastnosti komponenty InitialDirectory. Při zobrazení této komponenty dostaneme zadanou cestu a v této cestě můžeme dále vyhledávat potřebné soubory. Opět je zde možnost, že uživatel může překlikávat hiearchií a vybrat tak soubor z jiné než povolené cesty k souboru, proto i tato anomálie je ošetřená, stejně jako u výběru adresářů. Výběr souboru je rozšířen na možnost výběru i několika souborů zároveň, tím odpadá zdlouhavé vybírání souborů po jednom. Architektura přidávání hodnot je ale navržená pro přidání pouze jedné hodnoty, proto seznam souborů je zapsán



Obrázek 15: Výběr adresářů pomocí FolderBrowserDialog



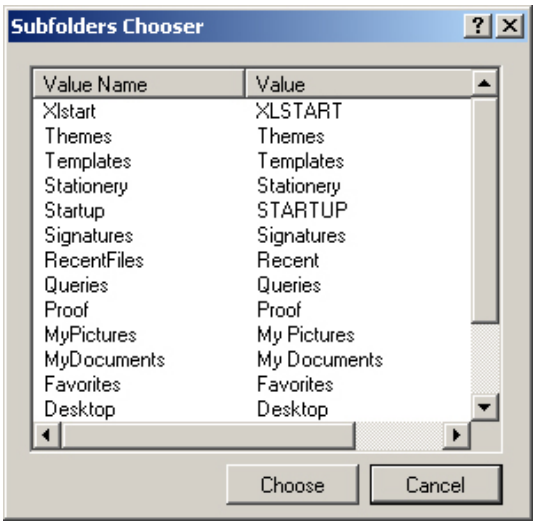
Obrázek 16: Manuální vkládání hodnot

jako jedna hodnota, která je rozdělena znakem ";", který se pak použije k rozkouskování hodnoty na jednotlivé části.

5.2.3 Manuální vkládání adresářů a souborů

Jak přidávání hodnot adresářů, tak i souborů lze realizovat i manuálně. Jedná se především o případy, kdy uživatel chce zadat název souboru s použitím wildcards. Aby při tomto úkonu nenastaly problémy se syntaxí, tak při najetí myší na TextBox pro vyplnění hodnoty se objeví nápověda, jak by měla být hodnota zadána, viz. obrázek 16. Pokud i přesto uživatel zadá neplatnou syntaxi, PW si s touto anomálií koneckonců poradí.

Další důvod proč existuje manuální vkládání je možnost přidávání Subfolders. Jedná se o složky, jejichž korektní název se nachází pod klíčem HKEY_CURRENT_USER\\Software\\Microsoft\\Office\\"Číslo verze MS Office"\\Common\\General. Pro dvě vlastnosti IncludeFolderTrees a FolderTreesToRemoveToResetToDefault je k dispozici na formuláři pro přidání hodnoty tlačítko, jak je vidět na obrázku 16, které zobrazí seznam hodnot výše zmíněného klíče, viz. obrázek 17. Aby uživatel použil správný název složky,



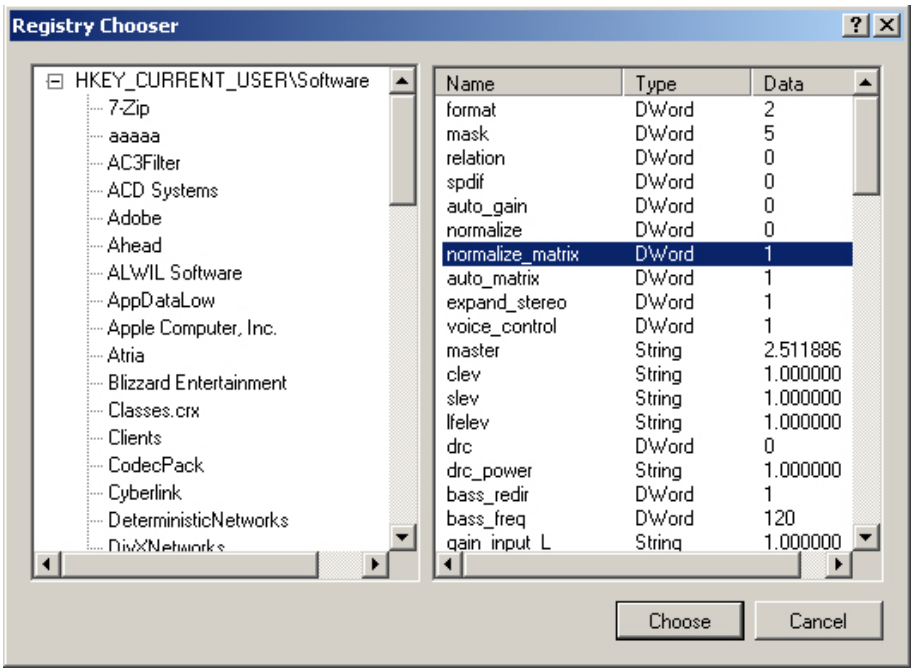
Obrázek 17: Seznam hodnot pro subfolder tokeny.

vybere ze seznamu název hodnoty, která správný název složky reprezentuje. Tento název složky se pak přidá k textu, který byl dosavaď napsán v komponentě TextBox. Uživatel by měl mít stále na paměti, že musí dodržovat správnou syntaxi pro tyto vlastnosti uvedenou v nápovědě po najetí myší na TextBox.

5.2.4 Klíče a hodnoty registru

Pro přidání klíče nebo hodnoty registru je vytvořen nový formulář. Vstupem pro tento formulář je kořenový klíč registru, ze kterého se budou vybírat podklíče a jejich hodnoty. Jako u PW tak i u NPW nelze zadat pouze kořenový klíč registru, ale aspoň jeden stupeň vnoření v hiearchii registrů. Na obrázku 18 je zobrazen formulář pro přidání hodnoty klíče registru, kde nalevo je komponenta TreeView se seznamem klíčů registů a napravo je komponenta ListView se seznamem hodnot aktivního klíče registru. Rozdíl oproti formuláři pro přidávání pouze klíče registru je ten, že seznam hodnot není povolen, tudíž seznam těchto hodnot je jenom informací, jaké hodnoty aktivní klíč registru obsahuje. Pro tento případ, kdy vybíráme i hodnoty, je tento seznam povolen a je možno z něj vybírat konrétní hodnoty pro konfigurační soubor. Když se nevybere hodnota klíče registru, použije se tzv. default hodnota.

Seznamem klíčů se dá proklikávat v hiearchii a vnořovat se hlouběji. Kořenový uzel tohoto seznamu klíčů slouží k překliknutí se v hiearchii výše. Tyto kořenové uzly nelze přidat jako klíč do konfiguračního souboru, slouží pouze k posunutí se v hiearchii výše. U některých klíčů se nebudeme moci dostat hlouběji v hiearchii, protože nebudeme mít povolen přístup (tento problém byl popsán v sekci vytváření datového souboru), tudíž je opět ošetřen tento krok pomocí bloků try/catch. Při použití tlačítka Choose předáme vybraný klíč nebo hodnotu k přidání do konfiguračního souboru.



Obrázek 18: Formulář pro přidávání hodnot klíče registru

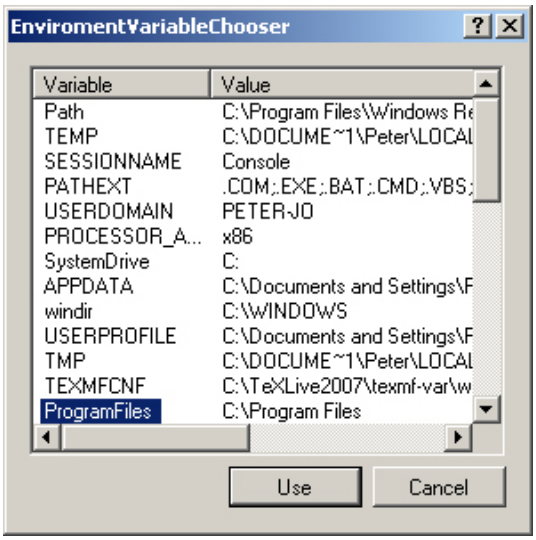
5.2.5 Proměnné prostředí

Hodnoty vlastnosti SubstituteEnvironmentVariables jsou proměnnými prostředí, které slouží pro uložení systémových nastavení či odkazů. O proměnných prostředí se dočtete také zde [19]. Každá proměnná pod sebou nese tedy specifickou hodnotu, která je jedinečná pro každý počítač či uživatelský účet. Hodnotami zapsanými pod touto vlastností se aplikaci MS Office Profile Wizard dává najevo, že se v registrech pod hodnotami, které mají typ REG_EXPAND_SZ, nahradí tato proměnná prostředí onou specifickou hodnotou.

Tato sekce se liší od těch předchozích tím, že nepoužívá pro přidávání hodnot formulář Add Item, ale přímo po kliknutí na Add tlačítko se zobrazí seznam všech proměnných prostředí a jejich hodnotami pro daný systém, viz. obrázek 19. Tento seznam je po vzhledové stránce téměř totožný se seznamem pro vkládání Subfolders, až samozřejmě na obsah, jehož hodnoty se berou z výstupu metody GetEnvironmentVariables třídy Enviroment. Tato metoda vrací objekt rozhraní IDictionary reprezentující negenerickou kolekci páru klíč/hodnota. Reprezentace tohoto páru je přes strukturu DictionaryEntry, což je rozdíl oproti klasické struktury KeyValuePart třídy Dictionary.

5.2.6 Konfigurační soubor aplikace

Pro stanovení cest k různým souborům a registrům existuje jejich zápis v konfiguračním souboru aplikace. Tím se ulehčuje správa nad těmito hodnotami. Podoba konfiguračního



Obrázek 19: Formulář pro vkládání proměnných prostředí

souboru je uvedená na výpisu 8.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key="HKCU" value="Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\Shell
        Folders" />
    <add key="officeHKCU10.0" value="Software\\Microsoft\\Office\\10.0\\Common\\General
        " />
    <add key="officeHKCU11.0" value="Software\\Microsoft\\Office\\11.0\\Common\\General
        " />
    <add key="officeHKCU12.0" value="Software\\Microsoft\\Office\\12.0\\Common\\General
        " />
   <add key="version" value="12.0" />
   <add key="product" value="Microsoft Office 12.0" />
  </appSettings>
</configuration>
```

Výpis 8: Konfigurační soubor aplikace uživatelské rozhraní pro MS Office Profile Wizard

Význam jednotlivých klíčů a jejich hodnot:

- HKCU - cesta ke klíči registru se seznamem souborových tokenů
- officeHKCU10 - cesta ke klíči registru se seznamem subsouborových tokenů pro MS Office verzi 10.0
- officeHKCU11 - cesta ke klíči registru se seznamem subsouborových tokenů pro MS Office verzi 11.0

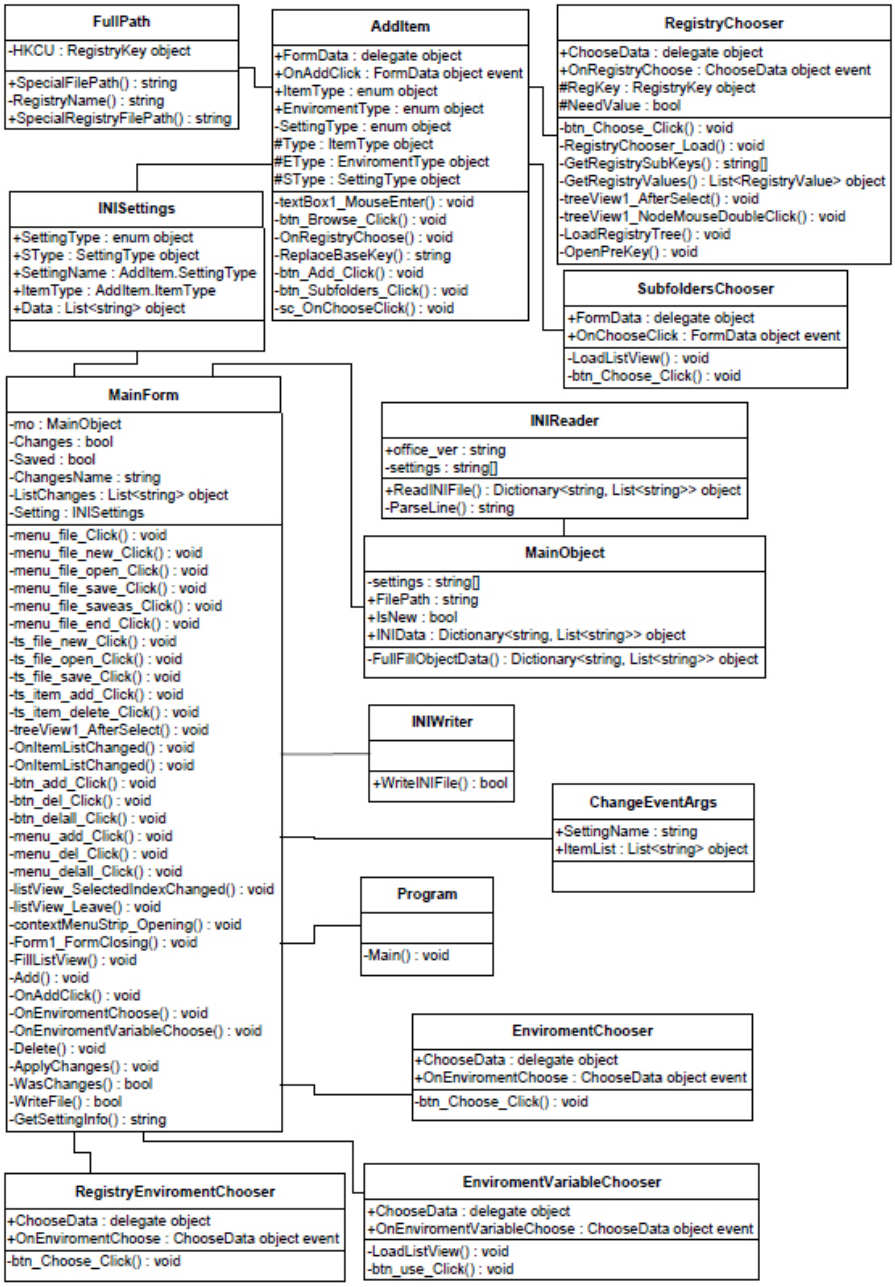
- officeHKCU12 - cesta ke klíči registru se seznamem subsouborových tokenů pro MS Office verzi 12.0
- version - zápis verze MS Office
- product - celý název verze MS Office

5.2.7 Třídní diagram aplikace

Seznam tříd a jejich rozvržení v aplikaci znázorňuje třídní diagram na obrázku 20.

5.3 Shrnutí aplikace uživatelského rozhraní

Výhodou této aplikace je, že uživatel může do konfiguračního souboru přidávat hodnoty, které se opravdu nacházeji na účtu uživatele (jak v registrech tak i na disku). Ulehčí tak práci s hledáním souborů a registrů a odpadá zdlouhavé vypisování absolutních cest k souboru nebo adresáři (respektive relativních cest vůči souborovému tokenu či kořenového klíče registru). Omezí počet anomálií v syntaxi na minimum. Díky těmto výhodám nenastane problém při parsování konfiguračního souboru aplikací MS Office Profile Wizard.



Obrázek 20: Třídní diagram aplikace uživatelské rozhraní pro MS Office Profile Wizard

6 Závěr

Důležitým faktem či doporučením při používání NPW, je zajištění, aby soubory, které jsou uvedeny při obnově z datového souboru, nebyly používány jiným programem. NPW pak nebude muset tyto výjimky přeskakovat a uživatel dosáhne 100% jistoty, že tyto soubory budou v pořádku obnoveny.

Na základě testů a jejich shrnutí lze konstatovat, že aplikace NPW nedopadla ve srovnání s aplikací PW od společnosti Microsoft vůbec špatně, ba dokonce ve většině případů lépe, ale jen co se týče výstupu těchto aplikací. Co se týče zpracování a implementace, nelze tyto dvě aplikace porovnat z důvodů neznalosti implementace MS Office Profile Wizard.

Uživatelské rozhraní dostatečně usnadňuje práci uživatele se sestavováním konfiguračního souboru apliakce MS Office Profile Wizard. Zvolené řešení a funkčnost uživatelského rozhraní dovoluje elegantně vyplňovat hodnoty jednotlivých nastavení.

7 Reference

[1] *Orbit s r.o.*,
http://www.orbit.cz/

[2] *Windows Server 2003 tutorial*,
http://www.visualwin.com/

[3] Microsoft TechNet, *Configuring Roaming User Profiles*,
http://technet.microsoft.com/en-us/library/cc738596(WS.10).aspx

[4] Dan Holme, Orin Thomas, *70-290 Managing and Maintaining a Windows Server 2003 Environment*,
Microsoft Corporation, 2004

[5] J. C. Mackin, Ian MacLean, *70-291 Implementing, Managing, and Maintaining a Windows Server 2003 Network Infrastructure*,
Microsoft Corporation, 2004

[6] Jill Spealman, Kurt Hudson, *70-294 Planning, Implementing, and Maintaining a Windows Server 2003 Active Directory Infrastructure*,
Microsoft Corporation, 2004

[7] *Microsoft Developer Network*,
http://msdn.microsoft.com/en-us/library/default.aspx

[8] Microsfot Office Online, *Office Profile Wizard*,
http://office.microsoft.com/en-us/ork2003/HA011513591033.aspx

[9] Microsoft TechNet, *Profile Wizard*,
http://technet.microsoft.com/en-us/library/cc750925.aspx

[10] *Command line switches for the Office Profile Wizard*,
http://support.microsoft.com/kb/281928/en-us?fr=1

[11] WindowsITPro, *Understanding Regedit.exe .reg files*,
http://www.windowsitpro.com/article/jsifaq/jsi-tip-8615-understanding-regedit-exe-reg-files-.aspx

[12] eHow, *How to Create Registry Files*,
http://www.ehow.com/how_5257035_create-registry-files.html

[13] Wikipedie, *Registr Windows*,
http://cs.wikipedia.org/wiki/Registr_Windows

[14] Technická podpora Microsoft, *Informace o registru systému Windows pro pokročilé uživatele*,
http://support.microsoft.com/kb/256986

[15] Tony Northup, Shawn Wildermuth, *70-536 .NET Framework 2.0 Application Development Foundation*,
Tony Northup, Shawn Wildermuth, Bill Ryan and Grand Masters, 2006

[16] The Code Project, *Converting Wildcards to Regexes*,
http://www.codeproject.com/KB/recipes/wildcardtoregex.aspx

[17] Living .NET..., *Process.Start() Quips*,
http://msmvps.com/blogs/manoj/archive/2004/10/23/16505.aspx

[18] CodeSource.net, *Reading and Writing Binary Files*,
http://www.codersource.net/microsoft-net/c-files/reading-and-writing-binary-files-in-c-net.aspx

[19] AdminXP.cz, *Proměnné prostředí (Environment Variables)*,
http://www.adminxp.cz/windowsvista/index.php?aid=235

A Obsah přiloženého CD

Na přiloženém CD můžete najít tuto adresářovou strukturu:

- Zdrojové kódy
  - aplikace New Profile Wizard
  - aplikace uživatelského rozhraní pro MS Office Profile Wizard (PWGUI)
- Spustitelné aplikace
  - aplikace MS Office Profile Wizard
  - aplikace New Profile Wizard
  - aplikace uživatelského rozhraní pro MS Office Profile Wizard (PWGUI)
- Text
  - obrázky použité v textu
  - text
  - zdrojový kód textu
- Programátorské příručky
  - programátorská příručka aplikace New Profile Wizard
  - programátorská příručka aplikace uživatelského rozhraní pro MS Office Profile Wizard (PWGUI)